Helsinki University of Technology
Publications in Telecommunications Software and Multimedia
Teknillisen korkeakoulun tietoliikenneohjelmistojen ja multimedian julkaisuja
Espoo 2004 TML-A10

NORDSEC 2004

Proceedings of the Ninth Nordic Workshop on Secure IT Systems – Encouraging Co-operation

November 4-5, 2004, Espoo, Finland

Sanna Liimatainen, Teemupekka Virtanen (eds.)

In cooperation with

Microsoft NOKIA ERICSSON

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY

Helsinki University of Technology
Publications in Telecommunications Software and Multimedia
Teknillisen korkeakoulun tietoliikenneohjelmistojen ja multimedian julkaisuja
Espoo 2004 TML-A10

NORDSEC 2004

Proceedings of the Ninth Nordic Workshop on Secure IT Systems – Encouraging Co-operation

November 4-5, 2004, Espoo, Finland

Sanna Liimatainen, Teemupekka Virtanen (eds.)

In cooperation with

Helsinki University of Technology
Department of Computer Science and Engineering
Telecommunications Software and Multimedia Laboratory

Teknillinen korkeakoulu
Tietotekniikan osasto
Tietoliikenneohjelmistojen ja multimedian laboratorio

Distribution:
Helsinki University of Technology
Telecommunications Software and Multimedia Laboratory
P.O.Box 5400
FIN-02015 HUT
FINLAND
URL: http://www.tml.hut.fi/

ISBN 951-22-7348-9
ISSN 1456-7911

Otamedia Oy
Espoo 2004

TABLE OF CONTENTS

PREFACE

It is my pleasure to welcome the participants of NordSec 2004 to Espoo. This is the ninth Nordsec conference and the second time it is arranged here. November is the darkest time of year and we all appreciate the light this conference brings to us.

Over the years there has been some variations in the focus of these conferences. We have defined this conference's focus as an easily accessible middle level scientific conference, targeted to post-graduate students. We have tried to create a simple, well organized conference, targeted to meet the needs of post-graduates and academics. Keeping the conference on budget does not imply any skimping on the academic quality of the papers or the review and publication process.

We are very happy with the content provided to the conference by the participants. There were 49 papers submitted out of which 19 papers were selected which means 39reviewed blindly by at least two members of the program committee. I like to thank the committee members and other reviewers for their work. The scientific community needs people who not only write but read and review, too.

Security is a very large discipline, to which almost any subject can be included into. In this conference we are studying solutions to one specific area. We try to improve security of information systems or improve security using information technology. We should keep in mind that outside this area are many other methods with which we together form the big picture. But as information technology becomes more important in the world, so does our work. Our area increases but in the same time it diffuses with other disciplines. As information technology becomes more common and the level of connectivity in our world expands, the need for security grows also. We can see this from different indicators, weather we are looking at national policies, RFC requirements or readers letters to newspapers, the need for security is there.

Now, let's start the conference. The authors and participants create a conference. We thank all the authors for the papers in this volume and the participants for their active participation.

Teemupekka Virtanen

PROGRAMME COMMITTEE

Teemupekka Virtanen, Program chair	Helsinki University of Technology
Mads Dam, Program co-chair	SICS & Royal Institute of Technology
Tuomas Aura	Mads Dam
Microsoft Research	SICS & Royal Institute of Technology
Ulfar Erlingsson	Pasi Eronen
Microsoft Research	Nokia Research Center
Simone Fischer-Huebner	Viiveke Fåk
Karlstad University	Norwegian University of Science and Technology
Dieter Gollman	Marko Helenius
Microsoft Research	Tampere University of Technology
Erland Jonsson	Hannu Kari
Chalmers University of Technology	Helsinki University of Technology
Svein J. Knapskog	Kaisa Nyberg
Norwegian University of Science and Technology	Nokia Research Center
Jussipekka Leiwo	Helger Lipmaa
Nanyang Technological University	Helsinki University of Technology
Christer Magnusson	Pekka Nikander
Stockholm University/RIT	Ericsson Finland
Hanne Riis	Nahid Shahmehri
Technical University of Denmark	Linköpings universitet
Camillo Särs	Teemupekka Virtanen
F-Secure Inc.	Helsinki University of Technology
Antti Ylä-Jääski	Louise Yngström
Helsinki University of Technology	Stockholm University/RIT

Additional Reviewers

Fredrik Björck, Leong Peng Chor, Claudiu Duma, Almut Herzog, Jesper Kampfeldt, Sven Laur, Adrian Leung, Leonardo Martucci, Valtteri Niemi, Robin Sharp, and Eduard Turcan.

Organizing Committee

Ursula Holmström, Helsinki University of Technology
Ansa Laakkonen, Helsinki University of Technology
Sanna Liimatainen, Helsinki University of Technology

PROGRAMME

Wednesday, 3 November 2004

17:00-20:00	Registration (doors close at 20:00)
17:00-23:00	Welcome Reception and Sauna, Computer Science Building, HUT, Espoo

Thursday, 4 November 2004

9:00-9:45	Registration
9:45-10:00	Welcome
10:00-11:00	**Session I**
	Daniel Cvrček *Dynamics of Reputation*
	G. Navarro, J. Garcia, J. A. Ortega-Ruiz *Chained and Delegable Authorization Tokens*
11:00-11:30	Coffee Break
11:30-12:30	**Session II**
	Pentti Tarvainen *Survey of the Survivability of IT Systems*
	Martin Boldt, Bengt Carlsson, Andreas Jacobsson *Exploring Spyware Effects*
12:30-13:30	Lunch
13:30-15:00	**Session III**
	Marco Giunti *Security Properties for Intrusion Detection*
	Karin Sallhammar, Svein J. Knapskog *Using Game Theory in Stochastic Models for Quantifying Security*
	Jussipekka Leiwo *A Technique for Modeling and Analysis of Common Criteria Security Environments and Security Objectives*
15:00-15:30	Coffee Break
15:30-17:00	**Session IV**
	Sandro Grech, Jani Nikkanen *A Security Analysis of Wi-Fi Protected Access™*
	Marko Schuba, Volker Gerstenberger, Paul Lahaije *Internet ID - Flexible Re-use of Mobile Phone Authentication Security for Service Access*
	Geir M. Køien *Principles for Cellular Access Security*
19:00	Conference Dinner, Haikaranpesä

Friday, 5 November 2004

9:00-10:00	Keynote speach: Urho Ilmonen, Director, Chief Security Officer, Nokia Oyj. *Security Requirements in Real Life*
10:00-10:30	**Session V**
	Sven Laur, Helger Lipmaa *On Private Similarity Search Protocols*
10:30-11:00	Coffee Break
11:00-12:30	**Session VI**
	Ragni R. Arnesen, Jerker Danielsson, Bjørn Nordlund *Carnival - An Application Framework for Enforcement of Privacy Policies*
	Gergely Tóth, Zoltán Hornák, Ferenc Vajda *Measuring Anonymity Revisited*
	Anand S. Gajparia, Chris J. Mitchell, Chan Yeob Yeun *The Location Information Preference Authority: Supporting User Privacy in Location Based Services*
12:30-13:30	Lunch
13:30-14:30	**Session VII**
	Lillian Kråkmo, Marie Elisabeth Gaup Moe *An Attack on the Stream Cipher Whiteout*
	Kaarle Ritvanen, Kaisa Nyberg *Upgrade of Bluetooth Encryption and Key Replay Attack*
14:30-15:00	Coffee Break
15:00-16:30	**Session VIII**
	J. Garcia, F. Autrel, J. Borrell, Y. Bouzida, S. Castillo, F. Cuppens, G. Navarro *Preventing Coordinated Attacks via Alert Correlation*
	Jarmo Mölsä *Mitigating DoS Attacks against the DNS with Dynamic TTL Values*
	Emmanuel Guiton *A Distributed Defence Mechanism against Low-Bandwidth TCP SYN Flooding Attacks*

This page is blank.

Dynamics of Reputation

Daniel Cvrček
University of Cambridge
Computer Laboratory
15 JJ Thomson Avenue, CB3 0FD Cambridge, UK
Email: Daniel.Cvrcek@cl.cam.ac.uk

Abstract— **To enforce security without user enrollment, trust (or reputation) systems were proposed to use experience as crucial information to support cooperation as well as for security enforcement mechanisms. However, use of trust introduces very difficult problems that still discourage from exploitation of trust for security mechanisms. The ability to change trust quickly and react effectively to changes in environment and user behaviour is profound for usability of mechanisms built on top of trust. Dempster-Shafer theory was proposed as a suitable theoretical model for trust computation. Here, we define general requirements for reputation dynamics and demonstrate that Dempster-Shafer theory properties are not as good as is widely thought. On the contrary, simple formulae work.**

Index Terms— **Reputation, trust, security, Dempster-Shafer theory, Dirac impulse, Sybil attack, combining evidence.**

I. Introduction

Many large-scale systems span a number of administrative domains. They imply economic and technology reasons hampering system-wide user enrollment and also prevent effective global infrastructure for flexible centralised administration to be established. Current security mechanisms, based on identity, cannot be used in such systems yet cooperation between diverse, autonomous entities is needed. The identity of entities may be unknown in such systems because pseudonymity, or even anonymity, is becoming a fundamental property of information systems [1], [2], [3]. The only information that can be used for any security decision is (partial) information about an principal's previous behaviour. Such systems are called *trust-based systems* [4], [5], [6], [7] or *reputation systems* to characterise their nature.

Each user may deploy tens or hundreds of pseudonyms and each pseudonym may be connected to transactions spread across the system. These facts imply the possibility of existence of a number of distinct trust values which are *valid* for one physical identity. We cannot, and do not even want to, prevent this due to preserving certain level of privacy. On the other side, we need to capture a user's behaviour as accurately as possible. Each system incorporating reputation/trust is based on two paradigms.

- **local trustworthiness evaluation** allows any entity (principal) to make use of behavioural evidence and determine the trustworthiness of other entities,
- **distribution of trust** makes it possible to inform other entities about these local results of trust evaluations.

There are systems not supporting mechanisms for trust propagation. Such systems cause high independence of trustworthiness of a digital identity in different parts of the system. It is a challenging task to find the limits of such systems with respect to privacy properties that may allow the existence of many digital identities of a principal. However, it seems obvious that such systems will be much more vulnerable to distributed attacks [8] as the ability to spread knowledge about malicious identities or ongoing attacks is limited. When we enhance trust-based model with indirect evidence (i.e. evidence observed by someone else) we may get a system with some small subspaces (trust domains) of partially mutually dependent trust values.

The next section briefly summarises some of the security requirements that make reputation systems distinct from identity-based systems. Section III describes the Dempster-Shafer theory of observation. Section IV contains experimental results gained with the use of Dempster combination rule and the arithmetic mean. The results are compared with stated security requirements.

II. Background

This paper focuses on the local use of trust computations. We believe that one of the most important properties of trust is its dynamics. A theoretical system model should allow rather precise parametrisation of trust value behaviour.

Any computation is based on a set of observations. The size of the set varies with time, not only during the initial phase of system deployment. It is significant that the number of relevant observations, implemented systems can store, is usually not defined with respect to security requirements but rather with feasibility requirements in mind. This fact is definitely not good for security but capabilities, such as memory size or computational power, are decisive.

In this context, there is another contradiction regarding the importance of old and new observations. While new observations are most important for immediate reaction on security threats imposed by a particular entity, old observations are significant for long-time cooperation. There may be frequent situations when a principal behaves correctly for a long time period but could then be attacked by a Trojan-Horse that

The work is funded by the SECURE project (IST-2001-32486), a part of the EU FET Global Computing initiative.

dramatically changes his behaviour for a short time. Long-term experience may allow faster renewal of the original trustworthiness after the attack is over.

The value of old observations is also important for the economics of security [1]. Emphasis on old observations may prevent easy, fast, and therefore cheap Sybil attacks. It means that there will be quite often a case when old observations are more important than newer ones. From an attacker's viewpoint, the long-term record is expensive to create, especially when trust transitivity (recommendations from other potentially corrupted principals) is weakened.

The last important requirement for trust computations is different sensitivity to positive and negative changes in behaviour. It is important for a model to allow radical negative changes of trust value in response to serious negative observations. Analysis of this requirement indicates that it is closely related to the difference between old and recent observations discussed above. This observation about long-term and short-term trust is akin to human perceptions of trust.

Summarising these requirements on trust dynamics:

1) the reaction to most recent evidence should be independent of the total amount of evidence:
   - there should be independence between security requirements and the computational potential of particular devices;
   - speed and degree of reaction should be specified independently on the number of observations, since this cannot often be estimated in advance.
2) the value of trust is proportional to the content (size) of the evidence set:
   - emphasis is given to the stability of the trust value, i.e. short excesses are not so important;
   - an alternative meaning of trust values can be derived from the economic cost of evidence set creation.
3) the actual trust value is kept in a range which allows maximum sensitivity to changes:
   - it is very hard to express the weight of a trust value that has not changed for a long time, regardless of observations being gathered.
4) positive/negative changes of trust are not symmetric:
   - negative changes – it may be necessary to react quickly in case of attacks;
   - positive changes – long-term positive observations should be retained.

You can see that e.g. items 1) and 2) are contradicting each other. It is therefore not clear, whether we can find a single function that would satisfy both these requirements or whether two functions must be used and their results combined. We propose to use the latter approach based on Dirac impulses and control theory.

A. Trust and Access Rights

Eventually, trust and risk become inputs for access control. A sufficient trustworthiness is what allows a principal to access data or use functionality of a system. Trust consists of two parts: an information (or certainty) value and a confidence value (proper trust). The information value expresses the amount of evidence/observations that were gathered and used for the trust value computation.

When you run an information system you distinguish between insiders and outsiders. An insider is a user that you personally know; you know his identity. He may be your employee so there is a contract that obliges you to pay him a salary and he (the principal) must abide your rules as stated in his contract. The principal is assigned a user account and a group that is associated with privileges in the information system. Recommendation systems or trust-based systems may enhance your ability to control access of insiders as well as for outsiders. You can punish the employee and you can revoke access rights from outsiders. The strength of reputation systems is that it is not necessary to enroll users into information system. It may lead to higher privacy but it also implies risks of forged identities.

With a reputation system you can either set parameters for trust evaluation in advance or you can let the system to evolve and adapt to changes. The latter requires some measurement mechanisms – risk analysis. The idea is to perform risk analysis (measuring security of system) continuously. However, you do not repeat the same computations all over again but contexts specifying subsets of evidence used for runs of risk analysis are changing. However, the amount of possible contexts may be so huge that it is impossible to evaluate risk for all of them. The system then may randomly select new ones and if there is a distributed system in place, interesting contexts (security threats) can be spread throughout the system[1].

B. Trust and Reputation

Many papers confuse the notions of trust and reputation. The use of the words seems to distinguish two groups of people working towards the same target – trust-driven security mechanisms. The first group comes from the area of ubiquitous computing and distributed system architecture for global computing is their concern. Here, the reasoning about trust is rather abstract [4], [5], [6], [7]. The second group is more application oriented, concerned with peer-to-peer systems. They tend to see trust as a new, interesting idea on how to enrich security mechanisms. The terminology is different for basically the same concept; while the former use *trust-based system* to describe the architecture, the latter define *reputation systems* to design mechanisms.

We believe that trust is a relation one-to-many while reputation expresses view of many parties on one user/agent. Trust is my subjective view of my communication partner while reputation is aggregated trust of many users. However, this distinction is not important for local processing that is targeted by this paper so we may use the notions interchangeably.

[1]The idea comes from immunology, when antibodies are created randomly, antibodies reacting to "self cells" are filtered out and the rest is set off into blood. When a reaction is encountered, antibodies of a given type are being produced in large amount to expunge "non-self cell".

III. A Theory for Trust Computation

Dempster-Shafer theory is perhaps the one most preferred for trust computation in ubiquitous and global computing. [9] presents an intuitive way of behaviour modelling by exploiting the theory of observation. A similar model is also used in the work of Jøsang et al [10], [11], [12].

We now give only a brief overview of basic terms. The more detailed description can be found in [9]. The authors start with a set $\mathcal{H} = \{h_1, ..., h_n\}$ of mutually exclusive and exhaustive hypotheses. They also assume a finite set $\mathcal{O}$ of observations and there must be a hypothesis h_i such that its encoding (probability) $\mu_{h_i}(ob) > 0$ for each $ob \in \mathcal{O}$. There is also defined an evidence space over $\mathcal{H}$ and $\mathcal{O}$ to be a tuple $(\mathcal{H}, \mathcal{O}, \mu_{h_1}, ..., \mu_{h_n})$.

What we need is to obtain a normalised value of observation *encoding*. This can be perceived as a level of the evidence contribution to a hypothesis. The authors use a simple method to compute it.

$$\omega(ob, h) = \frac{\mu_h(ob)}{\sum_{h' \in \mathcal{H}} \mu_{h'}(ob)} \tag{1}$$

The function $\omega(ob, h)$ expresses probability of a hypothesis h to happen as a consequence of an observation ob. The evidence is viewed as a *function* mapping a prior probability (before new evidence is encountered) of the hypothesis to a posterior probability. That is,

$$\mu_{i+1}(h) = \mu_i(h) \oplus \omega(ob, h) \tag{2}$$

where the operator $\oplus$ combines two probability distributions on $\mathcal{H}$. The operator is defined by the following equation, where H is a subset of hypotheses from $\mathcal{H}$ we are interested in.

$$(\mu_1 \oplus \mu_2)(H) = \frac{\sum_{h \in H} \mu_1(h)\mu_2(h)}{\sum_{h \in \mathcal{H}} \mu_1(h)\mu_2(h)} \tag{3}$$

Let us assume that the subset $H \subseteq \mathcal{H}$ contains all the hypotheses expressing positive behaviour, i.e. that a given user will behave the way that is desirable. The value obtained from (3) is then called trust.

A. Computation of Trust

We saw how a hypothesis probability evolves by adding normalised encodings of new observations in the previous section. However, all observations had the same weight regardless of their context – time, or any other information that may influence their value.

Zhong and Bhargava described two basic ways of computing trust in [13]. They introduced new mapping functions for posterior probabilities. Four particular function instances were defined and tested on several different types of users.

Trust update and trust analysis functions were defined. A trust update algorithm maintains current trust state and combines it with a new observation:

$$TS_1 = f_1(ob_1), TS_{i+1} = f_i(TS_i, ob_{i+1}), \tag{4}$$

A trust analysis function, on the other hand, stores a sequence of observations and uses them to compute new trust values. The practical implementation uses a *sliding window* (of size n in eqs. (5), (6)) to determine which observations should be used in computations.

$$TS_{1,n} = f_i(ob_1, .., ob_k), \;\; 1 \leq k \leq n-1 \tag{5}$$

$$TS_{k,n} = f_n(ob_{k-n+1}, ..., ob_k), \;\; k \geq n \tag{6}$$

where $TS_{k,n}$ represents the trust state evaluated from the interaction sequence of length n starting from ob_k (the latest observation).

The important issue is that both approaches conform to the combination rule (3) as defined above. It is realised by f_i being substituted by (3). The only difference is the number of summands in (3).

IV. Practical Tests

We have used simple scenarios for practical tests. As the first set of evidence we collected a set of e-mail messages with their SpamAssassin scoring. The second set contained a subset of emails with explicit user marking on whether the message is a spam or not (this set was much smaller – it contained under a dozen of events compared with several hundred email messages). Trust values are to express the probability of senders to be spammers or proper users. All experiments were performed on a basic set of over 500 messages from about 230 domains.

A. Dempster Combination Rule

We chose two subsets of messages received from particular domains and applied the Dempster combination rule on them. The result were discouraging as even simple tests demonstrated some negative properties.

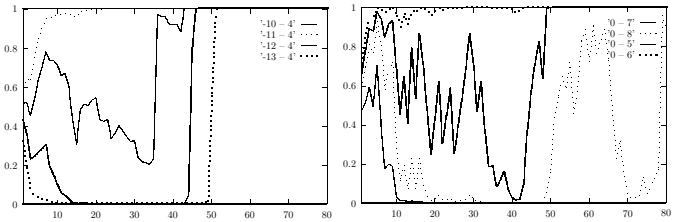
During the setup, one has to define evidence encoding functions. We have used simple linear function inside of total trust/distrust boundaries ($V_{trust}, V_{distrust}$).

$$\omega = \begin{cases} 0.01, & \text{if } score < V_{trust}; \\ 0.99, & \text{if } score > V_{distrust}; \\ 0.98\frac{V_{distrust} - score}{V_{distrust} - V_{trust}} + 0.01 & \text{otherwise.} \end{cases}$$

The domain of $score$ is a superset of all values $s \in \{V_{distrust}, V_{trust}\}$.

The graphs on fig. 1 demonstrate the results of trust computations for two e-mail domains with highest number of messages: *fit.vutbr.cz* (university) and *yahoo.com*. Parameters V_{trust} and $V_{distrust}$ are set manually to test thresholds where trustworthiness will change. Particular values are in the legends inside the graphs.

Authentic messages from *yahoo.com* are completed with a set of non-spam messages (from index 43) to test the time necessary for the change of trust values when a sudden change in behaviour occurs. Evidence encoding of the observations is created according to the rules above.



a) domain yahoo.com b) domain fit.vutbr.cz

Fig. 1. Dynamics of trust with Dempster combination rule

ev. encoding – ω	no of evidence
0.51	115
0.52	58
0.53	39
0,55	23
0.60	12

TABLE I

SPEED OF SATURATION ON 99 % TRUSTWORTHINESS

Even so, we can find two unpleasant properties. Trust values usually (unless V_{trust}, $V_{distrust}$ are carefully set for a particular evidence set) saturate at zero or hundred-percent trustworthiness. This situation is more thoroughly analysed in table I showing number of observations with a given encoding needed to saturate trust. Clearly, the Dempster combination rule works reasonably well in situations with a small amount of evidence and when two-value logic is of interest (e.g. when one needs to say whether a suspect is guilty or not). Neither of these assumptions is true in access control systems. We hope to have a large amount of evidence and we need a trust value allowing for fine-tuning of access control policies.

Concerning the reaction to a set of spam messages, fig. 2 demonstrates that the reaction is strongly dependent on the total amount of evidence. In fact, the delay between the attack detection (change in evidence encoding) and corresponding change of trust value can easily reach the time or number of observations related to the particular user/agent before the attack (the attack started with message indexed 51).

B. Improving Demspter Combination Rule

This property, saturation, is inherent to Dempster combination rule. We tried to solve this problem using an accumulator

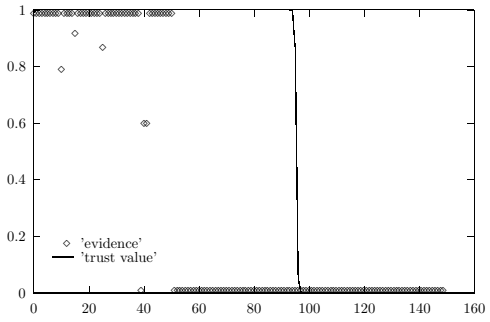


Fig. 2. Attack reaction delay with Dempster-rule

for a *surplus* trust. We were motivated by an analogy with the human perception of trust. Imagine you have known someone for quite some time. Unfortunately it happens that he makes a mistake that costs him some trust. However, this lost is usually short-term and after some time your long-term (accumulated) trust outweighs.

This effect may be modelled by setting a maximum and/or minimum level of trust T_{max}, T_{min}. We then create an *accumulator* T_{accum} of trust representing the effect of evidence that would cause trust to rise/drop below the stated boundaries.

Using (3) we obtain the following equation when one out of two hypothesis is being selected – h_1 expresses trust and h_2 distrust.

$$(\mu_{i-1} \oplus \mu_i)(h_1) = \frac{\mu_{i-1}(h_1)\mu_i(h_1)}{\mu_{i-1}(h_1)\mu_i(h_1)+\mu_{i-1}(h_2)\mu_i(h_2)} = \quad (7)$$

$$= \frac{\prod_{k=1..i}\omega_i(h_1)}{\prod_{k=1..i}\omega_i(h_1)+\prod_{k=1..i}\omega_i(h_2)} \quad (8)$$

The limiting condition (for high boundary) is

$$(\mu_1 \oplus \mu_2)(h_1) = T_{max} = \frac{\frac{\mu_1(h_1)\mu_2(h_1)}{T_{accum}}}{\frac{\mu_1(h_1)\mu_2(h_1)}{T_{accum}} + \mu_1(h_2)\mu_2(h_2)} \quad (9)$$

and

$$T_{accum} = \frac{(1-T_{max})\mu_1(h_1)\mu_2(h_1)}{T_{max}+\mu_1(h_2)\mu_2(h_2)} \quad (10)$$

The accumulator is empty when $T_{accum} \leq 1$. We have also defined the speed at which the accumulated trust can be released when the trust value changes rapidly. The accumulator has a positive influence on trust dynamics, giving instant response to attacks and controlled stability for long-term values (see fig. 3). Unfortunately, the saturation is a profound property of Dempster-Shafer theory which was created to give definitive answer yes or no (good or bad).

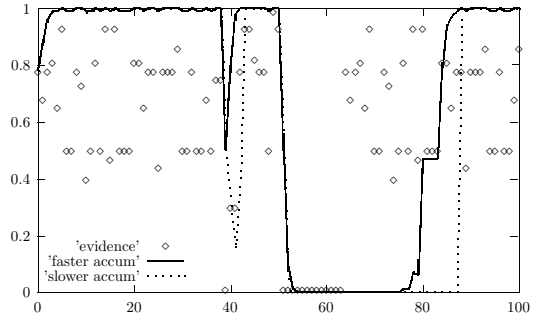


Fig. 3. Possible setting of reaction

C. Arithmetic Mean

The requirements described in section 2 led to the design of a second set of tests. Here, we returned to the simplest possible functions[2]. Beside this, we applied and tested dynamic update

[2]Although we used arithmetic mean as the simplest possible function, we have found that the consensus operator for dogmatic beliefs is computed in a very similar way, as described in [10].

of most of system parameters. The goal of this section is to demonstrate improvement in trust dynamics (stability and reaction to attacks) as a couple of refinements is applied.

The experiment settings that have to be made manually are very simple. We must state:

1) intervals within which the encoding function is monotonous – SpamAssassin scoring is monotonous on the whole domain of input values thus only one interval is identified;
2) whether the encoding function is decreasing or increasing function for all identified intervals;
3) and define evidence sources – we have two sources here – explicit marking spam/non-spam and SpamAssassin scoring.

a) Evidence Normalising: The following figure (fig. 4) shows three examples of encodings according to how the evidence is being normalised. When there is no explicit spam marking we obtain an evidence encoding with a very low average value in virtue of a much longer numerical interval representing non-spam messages. The dotted line demonstrates the influence of explicit marking (spam/non-spam). In this case, we got a better mid-value (0.5) but there is still a clearly visible impact of one, single, very low value of evidence on the whole aggregate.

The final experiment was to increase the sensitivity of the trust value in intervals with more evidence pieces. We have created five bands *on each side* from uncertainty (the gap between the lowest score of the marked spam and the highest score of marked non-spam). The boundaries of the bands were dynamically adjusted to contain approximately the same number of messages. Linear functions were used within the bands. This led to improved sensitivity of trust value as demonstrated by its increase towards 0.7, where 1 is absolute trust.

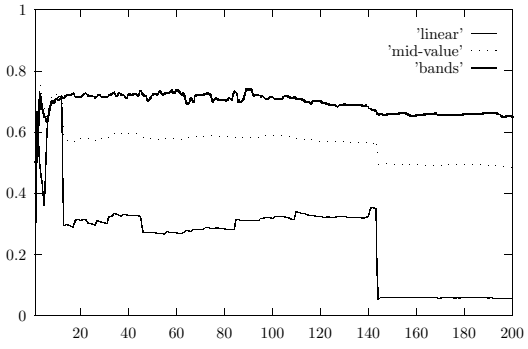


Fig. 4. Evolution with dynamic adjustment

While fig. 4 depicts evolution of trust in time, the graphs from fig. 5 show final reputation over email domains in the last experiment. All domains are at graph a) and domains with at least four messages are at graph b). (The arithmetic means for the graphs are 0.68 and 0.61 with variances 0.03 and 0.02 respectively.)

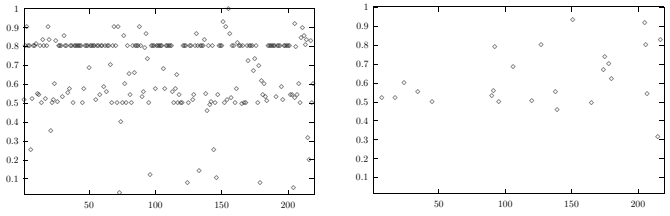


Fig. 5. Final trustworthiness

Initial trust value was set to 0.5 and we can see that most of the domains – their trustworthiness – lies between 0.5 and 0.8. Probably more interesting is the right graph where only one domain (yahoo.com) is significantly below neutral trust value.

The last graphs (fig. 6) demonstrate evolution of trust for the two domains with the largest number of messages. The beginning of the graph (value 0.5) is before receiving the first message. You can see that the trust value is very stable and does not significantly changes with new pieces of evidence.

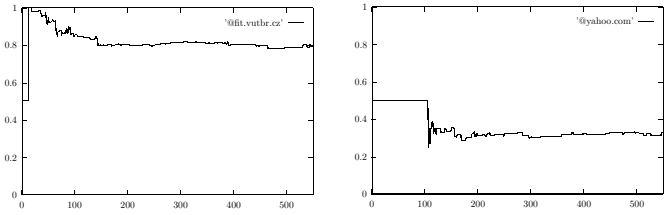


Fig. 6. Evolution with dynamic adjustment

We believe these results to prove trust computations to be versatile. Let us assume an example of an access control policy when a rule for granting access is defined as follows: aggregated trust computed over all granted accesses (messages are not marked as spam) is higher than 0.45 and trustworthiness of a particular domain must be above 0.4. To implement this policy, the system just uses the whole evidence set (without that pieces marked as spam) and all the evidence for a particular domain, and calls twice the same algorithm. The decision system is then implemented upon a simple table.

b) Sensitivity to attacks: When using the arithmetic mean the sensitivity to changes in evidence encoding decreases with the amount of evidence. The good news is that the sensitivity is easy to compute. We can define $T_{sens} = \frac{1}{\#evidence}$. This parameter can be used to normalise the impact of new evidence regardless of the number of observations in the evidence set used for trust computation. We may want any new evidence to have the same impact on the resulting trust value – we therefore set T_{impact}. The weight of this evidence should be:

$$weight = \frac{T_{impact}}{T_{sens}} = T_{impact} * \#evidence \qquad (11)$$

This is simple but does not allow the latest evidence to impact the trust value for more than one trust computation.

After experimenting with several approaches we recalled the *Dirac impulse* and its use for measuring system reactions

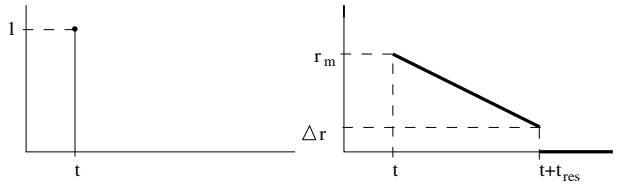


Fig. 7. System response on Dirac impuls

[14]. We found this to be a suitable solution for objective parametrisation of system reaction to attacks. We modelled the reaction as simple as possible, i.e. with linear function, see fig. 7. Three basic parameters represent maximum response (r_m), level of sensitivity (Δr) necessary for invocation of this correction mechanism, and duration of response (t_{res}). One can define two different responses for positive and negative changes with different parameters in the most complicated scenarios.

If there are several deviations in a row, we merely sum system responses and ensure the result fits between 0 and 1. Figures 8 and 9 demonstrate the dynamics of trust values with only negative system responses. Fig. 10 contains real data with several injected *attacks* represented by messages with indexes around 250.

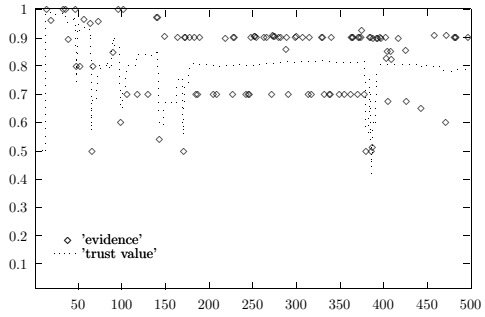


Fig. 8. Domain *fit.vutbr.cz* as is

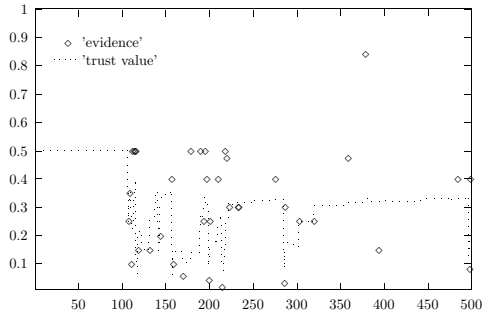


Fig. 9. Domain *yahoo.com* as is

The figures 8, 9, and 10 depict trust dynamics for the *fit.vutbr.cz* and *yahoo.com* domains. Parameters for the system response have been set as follows: $t_{res} = 2$, $\Delta r = 0.2$, and $r_{max} = 0.8$. Time is represented in number of messages rather than as real time and t_{res} is set low to demonstrate the ability to efficiently affect a trust value as a result of possible attacks. You can see that the trust value is now again nicely sensitive to short-time significant changes in behaviour while the long-trust value remains stable.

The Δr value is set explicitly. Real system could adjust Δr automatically according to results of risk analysis.

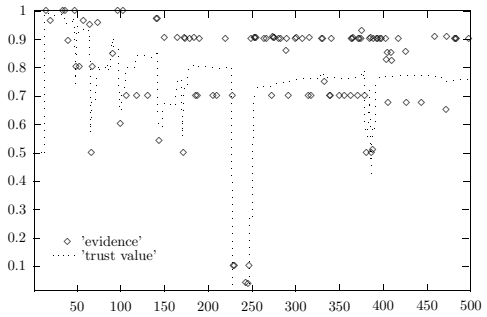


Fig. 10. Domain *fit.vutbr.cz* with short attack

V. Conclusions

We have shown how the Dempster Combination rule can be used in reputation systems. However, the experimental results point out that trust values very quickly saturate and it is impossible to use them for parameterised access control or any other security decision system. These results lead to a more precise definition of several basic requirements that should be fulfilled by relevant formulae.

Another important problem addressed is the possibility of getting useful results with very simple computations – represented here as arithmetic mean. We demonstrated suitability of this approach. The results are in conformance with the recent work of Audun Jøsang – the consensus operator, as defined, is just the weighted arithmetic mean.

However, special treatment of security issues is required. A possible solution was identified in combination with a separate definition of system response (normalised with Dirac impulse) for large deviations of behaviour. The resulting reputation is stable in time as well as sensitive to sudden attacks.

Dynamic recomputation of evidence encoding requires more computational resources but it ensures reasonable response in the face of large changes in the evidence domain when this is not known in advance.

Although we can easily find more functions or parameters that could be tested, we conclude that simple arithmetic functions ensure good functional behaviour when used in trust/reputation systems.

References

[1] A. Acquisti, R. Dingledine, and P. Syverson, "On the economics of anonymity," in *Financial Cryptography (FC '03), LNCS (forthcoming)*, 2003.

[2] R. Dingledine, N. Mathewson, and P. Syverson, "Reputation in privacy enhancing technologies," in *Proceedings of the 12th annual conference on Computers, freedom and privacy*, San Francisco, California, 2002.

[3] D. Cvrček and V. Matyáš, "Pseudonymity in the light of evidence-based trust," in *Proc. of the 12th Workshop on Security Protocols*, ser. LNCS (forthcoming). Cambridge, UK: Springer-Verlag, April 2004.

[4] A. Abdul-Rahman and S. Hailes, "Using recommendations for managing trust in distributed systems," in *IEEE Malaysia International Conference on Communication '97 (MICC'97)*. IEEE, 1997. [Online]. Available: citeseer.nj.nec.com/360414.html

[5] ——, "Supporting trust in virtual communities," in *Hawaii International Conference on System Sciences 33*, 2000, pp. 1769–1777. [Online]. Available: citeseer.nj.nec.com/article/abdul-rahman00supporting.html

[6] J. Bacon, K. Moody, and W. Yao, "Access control and trust in the use of widely distributed services," *Middleware 2001, Lecture Notes in Computer Science*, no. 2218, pp. 295–310, 2001.

[7] A. Jøsang, M. Daniel, and P. Vannoorenberghe, "Strategies forf combining conflicting dogmatic beliefs," 2003.

[8] J. Douceur, "The sybil attack," in *1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, ser. LNCS 2429. Springer-Verlag, 2002, pp. 251–260.

[9] J. Y. Helpern and R. Pucella, "A logic for reasoning about evidence," in *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI'03)*, 2003, pp. 297–304. [Online]. Available: http://www.cs.cornell.edu/riccardo/abstracts.html#uai03

[10] A. Jøsang, "The consensus operator for combining beliefs," *Artificial Intelligence Journal*, vol. 141, no. 1–2, pp. 157–170, 2002.

[11] A. Jøsang, M. Daniel, and P. Vannoorenberghe, "Strategies for combining conflicting dogmatic beliefs," in *Proceedings of the 6th International Conference on Information Fusion*, Cairns, July 2003.

[12] A. Jøsang, "Subjective evidential reasoning," in *Proc. of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2002)*, Annecy, France, July, 2002.

[13] Y. Zhong, Y. Lu, and B. Bhargava, "Dynamic trust production based on interaction sequence," Purdue University, Department of Computer Sciences, Purdue University, West Lafayette, IN 47907, USA, Tech. Rep. CSD-TR 03-006, 2003.

[14] R. Bracewell, *The Fourier Transform and Its Applications*. New-York: McGraw-Hill, 1999, ch. 5. The Impulse Symbol, pp. 69–97.

Chained and Delegable Authorization Tokens

G. Navarro, J. Garcia, J. A. Ortega-Ruiz
Dept. Informàtica, Universitat Autònoma de Barcelona,
Edifici Q, 08193 Bellaterra - Spain
Email: {gnavarro,jgarcia,jao}@ccd.uab.es

***Abstract*— In this paper we present an overview of an access control system based on tokens and delegable hash chains. This system takes advantage of hash chains in a similar way as the micropayment systems. On the other hand it uses an authorization infrastructure which allows to delegate authority and permissions by means of authorization certificates and delegation. The system is named CADAT (*Chained and Delegable Authorization Tokens*). We also describe one of the applications of CADAT, which is used as a token based access control for a secure mobile agent platform.**

***Index Terms*— Access Control, Authorization, Delegation, Hash Chains.**

I. Introduction

The access control and protection of computerized system is an active field of research, development and application. Lately there have been proposals for models, techniques and novel systems about access control in distributed systems. Normally they tend to look for the design of more flexible, scalable, and user-friendly systems while keeping a high degree of security.

Delegation, for example, introduces a lot of flexibility to a system, where the permission can be transfered or delegated between the users of the systems. Some of the first systems introducing delegation of authorization and the required infrastructure where KeyNote [1] and SPKI/SDSI (*Simple Public Key Infrastructure/Simple Distributed Secure Infrastructure*)[2]. Some authors refer to these systems as *trust-management*.

Another interesting field in access control is the use of access *tokens*. In these systems a user receives a token or ticket, which allows her to access a given resource or service. Token-based access control has been successfully used in several scenarios. For instance, Kerberos [3] is a popular system which makes use of access tickets.

An application, somehow similar to access tokens, are the micropayments. A micropayment system allows the payment of small amounts of money. Although micropayment systems have been around for a considerable time and have received some critics, they are still an interesting research field [4]. As a proof of concept, just consider companies that have appeared lately offering micropayment based systems, such as Peppercoin (`http://www.peppercoin.com`) or Bitpass (`http://www.bitpass.com`). One of the most interesting issues of the micropayment systems has been the introduction of hash chains to represent the transactions [5], [6], [7]. The use of hash chains allows to make the issuing of micropayment more flexible, since it substitutes computationally expensive cryptographic operations by simpler operations (hash functions).

There have been recently some propositions to use authorization infrastructures to implement micropayment systems. One of these proposals [8], introduces a model for using delegation in micropayment systems.

In this paper we present a token-based access control system. This system presents some advantages from micropayment systems and the delegation of authorizations. Our proposed system is called CADAT (*Chained and Delegable Authorization Tokens*). In CADAT the authorization tokens are constructed from a hash chain, which makes it faster and efficient to issue and verify tokens. Furthermore, we introduce the possibility to delegate token chains between users of the system in several ways, which adds flexibility to the system. CADAT is implemented by using SPKI/SDSI (Simple Public Key Infrastructure / Simple Distributed Secure Infrastructure), which provides the main framework to express authorizations and delegations. We also describe the application of CADAT as a token-based access control system in an specific scenario, a secure mobile agent application.

In Section II we introduce the main bases of the CADAT system. Section III introduces the SPKI/SDSI framework in relation to CADAT. In Section IV we show how to encode the hash chain based permissions into SPKI/SDSI certificates, providing the necessary modifications to SPKI/SDSI. Next, we show the main functionality of CADAT through an example in Section V. And in Section VI we discuss the application of CADAT in mobile agent systems. Finally, Section VII summarizes our conclusions and further work.

II. Hash chains and delegation

In this section we show how we can use delegation and hash chains together. It is based on the model and notation of [8]. We have simplified the model a little bit to make it more readable. The main idea is to consider the elements of a hash chain as permissions or authorizations.

We adopt the following notation and definitions:

K_A: public key of the user A.
sK_A: private key of the user A.
$\{m\}_{sK_A}$: digital signature of message m, signed by user A with the private key sK_A.
$(|K_B, p|)_{K_A}$ a direct delegation, where the user A, delegates permissions p to the user B. This relation can be established by means of an *authorization certificate*. At the moment we denote such a certificate as $\{|K_B, p|\}_{sK_A}$.

$h(m)$: one-way cryptographic hash function applied to m.
$[h^n(m), h^{n\ 1}(m), \ldots, h^1(m), h^0(m)]$: hash chain constructed from the initial message m (the seed of the hash chain). Normally, m may include information such as some random value (or nonce) to ensure its uniqueness.
$\mathcal{P}$: set of all the permissions managed by the system. There is an partial ordering in the set ($\preceq$) determined by the inclusion of one permission into another. For instance, $x \preceq y$, intuitively means that if a user holds permission y she also holds permission x. There is a least upper bound defined by the intersection ($\wedge$) of permissions, and a greatest lower bound defined by the union of ($\vee$). The order relation and the set of permissions form a *lattice*.
$\mathcal{P}_\mathcal{H}$: subset of permissions ($\mathcal{P}_\mathcal{H} \subseteq \mathcal{P}$), where each permissions is expressed as an element of a hash chain. The order relation between elements of $\mathcal{P}_\mathcal{H}$ is defined as: given $x, y \in \mathcal{P}_\mathcal{H}$, $x \preceq y$ in relation to principal P, if and only if P, knowing y, can easily determine some x and i, such that $h^i(x) = y$[8]. It means that principal P can find x and i without having to invert the hash function.

Following with these notation we can express the reduction rules of SPKI/SDSI [2], given the set of permissions $\mathcal{P}_\mathcal{H}$, in the following way:

$$\frac{(|K_B, x|)_{K_A}; y \preceq x}{(|K_B, y|)_{K_A}} \tag{1}$$

$$\frac{(|K_C, x|)_{K_B}; (|K_B, y|)_{K_A}}{(|K_C, x \wedge y|)_{K_A}} \tag{2}$$

The use of hash chain elements as permissions has several advantages, such as the possibility to issue specific permissions without having to issue any kind of certificate. We can also delegate (or transfer) parts of the hash chain to other principals.

To show the use of hash chains with delegation, we consider the following example. There are three users Alice, Bob and Carol, with their own public keys K_A, K_B, K_C. We assume that Alice is some authorization authority. Alice generates the following hash chain from the initial seed m: $[h^5(m)$, $h^4(m)$, $h^3(m)$,$h^2(m)$,$h^1(m)]$, where each element of the chain corresponds to an specific permission.

Alice can issue the following certificate:

$$\{|K_B, h^5(m)|\}_{sK_A} \tag{3}$$

With this certificate, Alice delegates $h^5(m)$ to Bob, who now, holds the permission $h^5(m)$. If Alice wants to delegate permission $h^4(m)$ to Bob, there is no need for Alice to issue another certificate. Alice just has to make public the value $h^4(m)$. Once it is public, Bob becomes automatically the holder of the permission $h^4(m)$. Bob is in the position of demonstrating that he holds such a permission because of certificate (3). In a same way, Alice can make public the successive permission, for instance, $h^3(m)$.

In a more general formulation, if Bob has a certificate, which directly grants him the permission $h^i(m)$, and Alice makes public the value $h^j(m)$, where $i > j$, Bob becomes the holder of permissions $h^i(m), \ldots, h^j(m)$. Given the reduction rule (1), we have that $(|K_B, h^i(m)|)_{sK_A}$, and $h^j(m) \preceq h^i(m)$, so $(|K_B, h^j(m)|)_{sK_A}$.

Another important issue is the introduction of delegation. Bob could delegate part of the permissions to another principal. Following with the example, Bob is in possession of $h^5(m)$, $h^4(m)$ and $h^3(m)$. He can delegate the last permission to Carol by issuing the following certificate:

$$\{|K_C, h^3(m)|\}_{sK_B} \tag{4}$$

Bob delegates $h^3(m)$ to Carol. If Alice makes public the value $h^2(m)$, Carol can demonstrate that she is in possession of such permission, because of the certificates (3) and (4). It is important to note that in no case, Carol can demonstrate she holds permissions $h^5(m)$ and $h^4(m)$. She can only receive permission of the chain which have and index lower or equal to $h^3(m)$.

We have seen how hash chains have important advantages when combining them with authorization or *trust management* systems. In the following section we show how this model can be implemented by using SPKI/SDSI. In [9], the authors show how to implement a similar system in KeyNote.

III. SPKI/SDSI AUTHORIZATION CERTIFICATES

SPKI/SDSI (Simple Public Key Infrastructure/Simple Distributed Secure Infrastructure), is a certificate-based authorization infrastructure, which can express authorizations and its delegation by means of *authorization certificates*. It also provides a distributed name system based on local names, by means of *name certificates*.

In SPKI/SDSI each principal has a pair of cryptographic keys, and its represented by its public key. In other words, we can say that in SPKI/SDSI each principal is its public key, and it is represented by the key or a hash of the key. An authorization certificate binds a public key with an authorization or permission. This way we can avoid the indirection present in traditional PKIs, where there is a binding between a public key and a global identifier (or distinguished name) and another one between the identifier and the authorization (permission or attribute). We can denote an authorization certificate as:

$$(I, S, tag, p, V) \tag{5}$$

Where:

I: *issuer*. The principal granting the authorization.
S: *subject*. The principal receiving the authorization.
tag: *authorization tag*. The specific authorization being granted by the certificate.
p: *delegation bit*. If it is active, the subject of the certificate can further delegate the authorization (or a subset) to another principal.
V: *validity specification*. It includes the validity time range (not-after and not-before) of the certificate and other possible conditions (currently online tests for revocation, revalidation and one-time revalidation).
$Comment$: although we do not show it in the notation, the certificates include a field of arbitrary information.

As one can see, the authorization certificates of SPKI/SDSI allows us to easily express the authorization certificates commented in Section II. For instance, certificate $\{|K_B, p|\}_{sK_A}$ can be expressed in SPKI/SDSI, for a given validity time V as: $(K_A, K_B, p, 1, V)$. Note that the delegation bit is active, so K_B can further delegate the authorization, K_A could avoid this by setting the delegation bit to 0 if needed.

IV. Hash chains as SPKI/SDSI authorizations

As we have seen permissions are expressed in SPKI/SDSI as the element *tag*. The format used by SPKI/SDSI to represent all the information is S-expression. In order to make it easy to process the authorizations, we include the index of the hash chain component , and an identifier of the chain in the authorization *tag*. This way a permission $p \in \mathcal{P}_{\mathcal{H}}$ will have the following format:

$$p = (cid, i, h^i(m)) \tag{6}$$

Where i is the index of the element in the chain and cid is the identifier of the whole hash chain. All the elements of the same hash chain have the same cid, which is a random bit string big enough to ensure its uniqueness. The seed, m includes additional information such as the public key of the principal that generated the chain, etc. The hash on m also includes i and cid in each step, making it more difficult to forge it. We do not get into details on how is the concrete information in m and how is the hash exactly computed, to keep the notation more readable, and because it may depend in the specific application of the permission. The reader may refer to [5], [6], [7] for specific ways to build such a hash chain.

SPKI/SDSI provides a certificate chain discovery algorithm to find authorization proofs in delegation networks [10]. This algorithm is based on basic intersection operations to provide certificate reduction rules. In [2] the intersection of the tag element is determined by the operation *AIntersect*.

In order to represent the authorization (6) in a SPKI/SDSI tag, we consider two alternatives. These alternatives are based on the need of verifying the hash of the permission in the tag intersection operation. When we have to intersect two permissions with the hash elements $h^i(m)$ and $h^j(m)$, if $i \geq j$ the resulting tag will be the $h^i(m)$. The intersection operation can also verify that $(h^j)^x(m) = h^i(m)$ for some x. This verification allows to immediately dismiss forged or erroneous permissions. If the verification is not carried by the intersection operation, it is important to note that the verification has to be done afterwards to validate an authorization proof.

A. Tag intersection without hash verification

In this case we can encode the permissions by using existing SPKI/SDSI structures. For example, a straightforward representation of the a permission like (6) in SPKI/SDSI S-expression format can be:

```
(tag
    (h-chain-id |123456789|)
    (h-chain-index (* range numeric ge 7)))
    (h-val (hash
       md5 |899b786bf7dfad58aa3844f2489aa5bf|))
```

Where `h-chain-id` is the identifier of the hash chain, *h-chain-index* is the index of the element in the hash chain and *h-val* is the value of the hash itself. The most important element is the index, which is expressed as a numeric range that will intersect with a range greater or equal to 7 in this case.

The only problem of the previous example is that if we include the value $h^7(m)$ in the *tag*, the intersection operation will not properly work. What we do is to introduce a little modification. The value `h-val` has to be treated in a different way. Thus, we put the *h-val* as the comment of the certificate. The certificate may look something like this:

```
(cert
  (issuer
    (hash
     md5 |1ac461a2e12a77ad54c67128b5060f28|))
  (subject
    (hash
     md5 |b0a746de2d5f6038e49a87c9c826bf4e|))
  (tag
    (h-chain-id |123456789|)
    (h-chain-index (* range numeric ge 7)))
  (comment
    (h-val
      (hash
       md5 |899b786bf7dfad58aa3844f2489aa5bf|)))
  (not-after "2004-01-01_00:00:00")
  (not-before "2005-01-01_00:00:00")
)
```

This allows us to use the SPKI/SDSI decision engine directly, without any lose of information. The main disadvantage of this approach is that in order to verify an authorization proof, the verifier needs to do an additional operation: verify the integrity of the hash chain (or subchain).

B. Tag intersection with hash verification

In this case, it is necessary to redefine the tag intersection operation for authorizations corresponding to elements of a hash chain. To do this we introduce a new kind of *tag*, the `<tag-hash-auth>`. The BNF definition of this new tag, according to the SPKI/SDSI tag definition [11] is given in Figure 1.

```
<tag>:: <tag-star> | "(" "tag" <tag-expr>  ")";
<tag-star>:: "(" "tag" "(*)" ")" ;
<tag-expr>:: <simple-tag> | <tag-set> |
             <tag-string> | <tag-hash-auth>;
<tag-hash-auth>:: "(" "hash-auth" <chain-id>
                  <chain-index> <hash>")";
<chain-index>:: "(" "chain-index" <decimal> ")";
<chain-id>:: "(" "chain-id" <byte-string> ")";
```

Fig. 1. Definitionn of `<tag-hash-auth>`.

We also introduce a new intersection operation: *HCAIntersect* (*Hash Chained Authorization Intersection*). The intersection of two `tag`s representing permissions of $\mathcal{P}_{\mathcal{H}}$ will result in the tag with the greatest hash chain index, if the hash chain identifier is the same and we can verify the hash values. For example, given the following `tag`s:

```
(tag
  (hash-auth
    (hchain-id |lksjfSDFIsdfkj0sndKIShfoMSKJSD|)
    (hchain-index 14)
    (hash md5 |899b786bf7dfad58aa3844f2489aa5bf|)))
```

```
(tag
  (hash-auth
    (hchain-id |lksjfSDFIsdfkj0sndKIShfoMSKJSD|)
    (hchain-index 15)
    (hash md5 |d52885e0c4bc097f6ba3b4622e147c30|)))
```

Its intersection (HCAIntersect) will be equal to the second tag, because the identifier is equal and the index of the second tag is greater that the first one. And we can verify the hash value of the tag. Note that the MD5 of the first value is equal to the second one.

We show the algorithm used by HCAIntersect with hash verification (*HCAIntersect full algorithm*).

Algorithm 1: HCAIntersect full algorithm

input : $p = (id_p, i, h^i(m)_p)$, $q = (id_q, j, h^j(m)_q)$, such that $p, q \in \mathcal{P}_\mathcal{H}$
output: r such that $r = HCAIntersect(p, q)$
begin
  if $id_p \neq id_q$ **then** $r \leftarrow NULL$;
  if $i \geq j$ **then**
    if *verifyHashSubChain(p,q)* **then** $r \leftarrow p$;
    else $r \leftarrow NULL$;
  end
  else
    if *verifyHashSubChain(q,p)* **then** $r \leftarrow q$;
    else $r \leftarrow NULL$;
  end
end

Algorithm 2: verifyHashSubChain function

input : $p = (id_p, i, h^i(m_p))$, $q = (id_q, j, h^j(m_q))$, where $i \geq j$
output: $res = true$ if $h^i(m)$ and $h^j(m)$ belong to the same hash chain, $res = false$ otherwise
begin
  $res \leftarrow false$;
  $aux \leftarrow h^j(m_p)$;
  for $x \in [(j-1)..i]$ **do**
    if $aux = h^i(m_q)$ **then** $res \leftarrow true$;
    $aux \leftarrow h(aux)$;
  end
end

The implementation of the hash chain elements as authorizations and the HCAIntersect algorithms, has been done in Java using the JSDSI [12] library. JSDSI is an open source implementation of SPKI/SDSI in Java, which has lately been under active development. The implementation of the new *tag* and the algorithm just represented a few modifications of JSDSI.

V. Application to access tokens: CADAT

The main motivation for the design of CADAT is its application as a token based access control system. In this section we show the basic functionality of CADAT through an example. We consider an scenario where news agencies allow access to the news databases to their clients. The access is controlled by tokens, that is, each time a user accesses (or reads) a new he needs to issue a token (depending on the case a given new may require several tokens).

For example the headquarters for the news agency AcmeNews wants to issue 9 access tokens (note that normally token chains will be much larger) to the user Alice, so she can access all the agencies worldwide (AcmeNews-SudAfrica, AcmeNews-India, ...). To do that, AcmeNews issues a contract to Alice authorizing her to use 10 access tokens *acme* (the first token is not used as an access token) for a given validity specification V_0. This contract is represented as an SPKI/SDSI authorization certificate.

$$(AcmeNews, Alice, auth_{star}, p = true, V_0) \quad (7)$$

Where $auth_{star}$ corresponds to: $(acmeID, 10, *)$. This is a `<tag-hash-auth>`, with the hash value equal to "*". It stands for an especial tag symbol used in SPKI/SDSI, which intersects with any kind of character string.

We denote this first certificate as `chain-contract-cert` or *chain contract certificate*, because it establishes a contract which allows Alice to demonstrate that she has been authorized by AcmeNews to use 9 tokens $acmeID$.

Now, Alice can generate the hash chain with 10 elements:

$$[(acmeID, 1, h^1(m)), (acmeID, 2, h^2(m)), \ldots, (acmeID, 10, h^{10}(m))] \quad (8)$$

The initial message or seed m will normally include information from the certificate (7), or specific information shared by AcmeNews and Alice.

Suppose that Alice goes to the AcmeNews-Antartida agency to look for news regarding the habits of the *Aptenodytes forsteri*[1]. To do that, Alice, establishes and initial contract with AcmeNews-Antartida with the following authorization certificate:

$$(Alice, AcmeNews-Antartida, auth_{10}, p = true, V_1) \quad (9)$$

Where $auth_{10} = (acmeID, 10, h^{10}(m))$. This certificate is denoted as a *token-contract-cert* or *token contract certificate*. It allows Alice to establish against AcmeNews-Antartida that she is in the position of spending 10 tokens acme. The agency AcmeNews-Antartida can verify such a claim by means of the certificates (7) and (9) by using the SPKI/SDSI decision engine.

Once, the *token-contract-cert* has been issued, Alice can begin to spend tokens to access the resources of AcmeNews-Antartida. For example, Alice can send the value $auth_9 = (acmeID, 9, h^9(m))$ and $auth_8 = (acmeID, 8, h^8(m))$. AcmeNews-Antartida can verify that $h(h^9(m)) = h^{10}(m)$ and that $h(h^8(m)) = h^9(m)$, and together with the certificates (7) and (9) can allow the requested access to Alice. By making the elements of the chain public, Alice has implicitly delegated the tokens to AcmeNews-Antartida based on the initial contract *token-contract-cert* (9).

[1] Also known as emperor penguin.

A. *Delegation of* token-contract-cert

Imagine now, that AcmeNews-Antartida decides to transfer the initial contract with Alice, *token-contract-cert* (9) to AcmeNews-Zoology because Alice needs to access some records of the zoology department. To do that AcmeNews-Antartida, issues a new *token-contract-cert* to AcmeNews-Zoology with the last token that it has received from Alice:

$$(AcmeNews - Antardtida, AcmeNewsZoology, auth_8, p = true, V_2) \tag{10}$$

AcmeNews-Zoology can verify together with the *token-contract-cert* (9) and the *chain-contract-cert* (7), that Alice is properly authorized to spend 7 tokens acme. Now Alice issues the next token $auth_7$, which is accepted by AcmeNews-Zoology, who can make all the pertinent verifications.

By means of the *token-contract-cert* (10), AcmeNews-Antartida has been able to delegate part of its initial contract with Alice to AcmeNews-Zoology. Normally this delegation can be transparent to Alice, allowing AcmeNews to easily subcontract services.

Figure 2 shows both *token-contract-cert*s issued, and the tokens delivered by Alice.

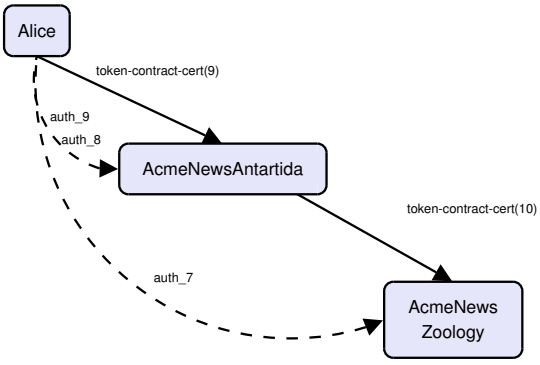


Fig. 2. *token-contract-cert* delegation example.

B. *Delegation of* chain-contract-cert

There is another contract delegation possible, the delegation of the *chain-contract-cert* by Alice. Imagine that Alice decides to transfer (delegate) the rest of the tokens to her colleague Bob, so he can use them. To do that, she just has to delegate the *chain-contract-cert* (7) that she received directly from AcmeNews by issuing the following certificate:

$$(Alice, Bob, auth7, p = true, V_3) \tag{11}$$

It authorizes Bob to spend the remaining 6 tokens from the chain. It is important to note that Alice has to let Bob know somehow the hash chain (8) or the initial seed m. This value could, for example, be encrypted in the comment field of the certificate.

Now Bob can issue the value $auth_6$. This token will be accepted by AcmeNews-Zoology, who can verify that Bob is authorized to issue it due to the *chain-contract-cert* (7) and (11). Figure 3 shows the *chain-contract-cert* and the tokens issued by Alice and Bob.

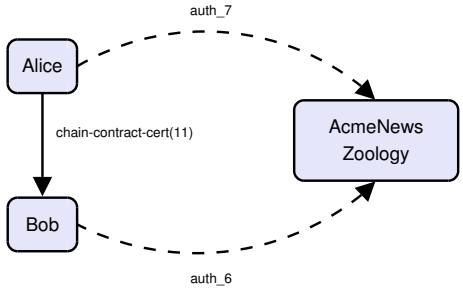


Fig. 3. *chain-contract-cert* delegation example.

C. *Comments on delegation*

In the previous section we have introduced both the *chain-contract-cert* and the *token-contract-cert*. Although both certificates look very similar, their differentiation is a key concept determined by the use of the contract. Summarizing:

> *chain-contract-cert*: allows to delegate the hash chain o a subchain to a principal. The principal receiving the contract will be the one using the tokens to access or use a given service. An special case is the first certificate, which establishes the trust that the authority (AcmeNews) places in a given principal (Alice).
> *token-contract-cert*: this certificate also allows to delegate the hash chain or a subchain, to a principal. But in this case, the principal receiving the contract is the consumer of the tokens. This situation is specially interesting because allows the outsourcing of services from one organization to another in a transparent way.

Note that each contract certificate has an specific validity specification, which allows for a fine-grained control of the delegations.

As we have seen, the ability of delegating the permissions as hash chains (or subchains) in two different ways introduces one of the key issues of CADAT. As another way to view the two types of delegation supported by the system, we can say that there is a client-side delegation (*chain-contract-cert*) and a server-side delegation (*token-contract-cert*).

Although we have used SPKI/SDSI as the underlying infrastructure, a similar approach could be used with other trust management technologies such as KeyNote. And in general, in other infrastructures supporting delegation, for instance the last X.509 specification describes extensions to support the delegation of attributes.

There may be the possibility of double spending. That is, a user that uses a token more than once. In order to avoid it, we consider that the principal making the original contract keeps track of the tokens used by all its contractors. In the above example, AcmeNews is the authority granting the tokens to Alice, we can say that AcmeNews is the authority for *acme tokens*. In order to accept tokens AcmeNews-Antartida and AcmeNews-Zoology, check the corresponding contract certificates. They will only accept acme tokens if the authorization root for those tokens is AcmeNews. The system should provide the ability for AcmeNews to keep track of spent tokens, for example, by means of online notifications.

VI. Application of CADAT

CADAT attempts to be a generic system, which can be applied in several scenarios. Given its nature, it can be easily used as a micropayment system, introducing the ability to delegate payment tokens. In [8], the authors describe a micropayment system implemented in KeyNote that makes use of what we call delegation of *token-contract-cert*, but does not uses the *chain-contract-cert* delegation. CADAT can be seen as an extension of this system.

But the CADAT system can also be used as a generic token based system. One of the current implementations of CADAT is as a token-based access control in a secure mobile agent platform. Access control in mobile agent systems involves lots of security problems. One of them is the fact that if mobile agents are able to perform digital signatures, either their private key is visible to the underlying platform, or the agent has to use a third party cryptographic proxy.

There are some alternatives as [13], which allows to delegate authorizations to a hash of the agent's code, but presents some constraints in the system. We have implemented CADAT on top of a secure mobile agent system, which at the same time makes use of the JADE (Java Agent Development Framework) framework[14], which supports the main FIPA (Foundation for Intelligent Agents)[15] specifications. JADE is a popular open source generic agent platform developed in Java, which lacks support for mobility and security. The implementation is done within the MARISM-A project, which attempts to bring mobility and several security solutions to the JADE platform. The mobility implemented in top of JADE is described in [16], and some existing security solutions in [17]. In the context of MARISM-A, CADAT provides a token based authorization mechanism specially interesting for mobile agents.

Figure 4 outlines the scheme used in the mobile agent application. A user Alice establishes a *chain-contract-cert* with a token authority, which will allow her to generate the specified number of tokens for her mobile agents. In order to access a resource in a given agent platform, Alice establishes a *token-contract-cert* with the platform. Tokens are made public by Alice when one of its agents has to access the platform resource. The platform verifies all the contracts and the tokens published by Alice to allow the agent to access the requested service or resource.

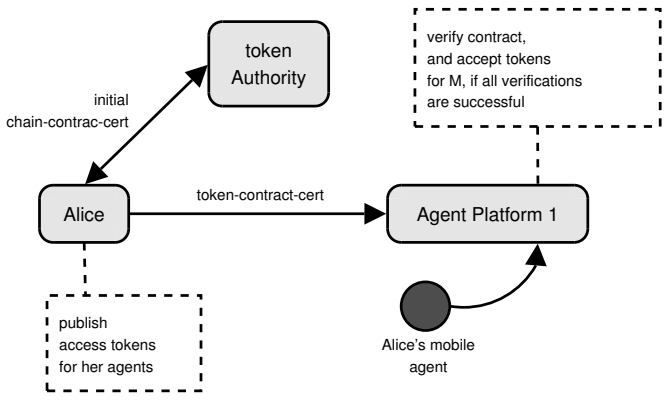


Fig. 4. CADAT in mobile agent applications.

In this case, the initial seed of the hash chain includes a hash of the agent code, which is used by the agent platform to authenticate the mobile agent. When the agent accesses the platform's resources, it does not have to perform any kind of cryptographic operation. The agent itself, does not need to carry any kind of sensitive information such as cryptographic keys or even the access tokens.

CADAT provides an extension of more general authorization frameworks such as [13] for mobile agent applications. It is specially suited for open environments, where authorizations are granted by a previous agreement with some application authority. It avoids common problems such as the maintenance of complex policies and certificate based infrastructures. The use of delegation also provides a lot of flexibility. An agent platform can outsource services or resources from other platforms by delegating the *token-contract-cert*, and the user can transfer part of the tokens to other users by delegating the *chain-contract-cert*.

VII. Conclusions and further work

In this paper we have presented the ideas behind the CADAT system. We have described its main base and it use and general functionality. The main idea behind CADAT is the use of elements of a hash chain as authorization tokens, the possibility to delegate those tokens in different ways.

One of the applications of CADAT is its use as a token based access control system in a secure mobile agent platform. Mobile agents do not need to carry any kind of sensible information as cryptographic keys. Furthermore the agents do not even have to perform costly cryptographic operations. The delegation introduced in CADAT allows a user to delegate access tokens to other users, and platform agencies (or servers in general) to outsource services and resources to other platforms (or server) in a way that is transparent to the user.

We are currently working on the improvement of the prototype implementation of CADAT in the mobile agent application. In relation to SPKI/SDSI, we have not discussed the implications of the use of threshold certificates and name certificates, which could add extra value to the system. We are also considering other possible implementations of CADAT such as a generic micropayment system for web services.

Acknowledgment

This work has been partially funded by the Spanish Government Commission CICYT, through its grants TIC2003-02041 and TIC2001-5108-E, and the Catalan Government Department DURSI, with its grant 2001SGR-219.

References

[1] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, "The KeyNote Trust Management System," RFC 2704, IETF, Sept. 1999.
[2] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, "SPKI Certificate Theory," RFC 2693, IETF, September 1999.
[3] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications*, pp. 33–38, 1994.
[4] M. Lesk, "Micropayments: An Idea Whose Time Has Passed Twice?" *IEEE Security & Privacy*, January/February 2004.
[5] R. Anderson, H. Manifavas, and C. Shutherland, "Netcard - a practical electronic cash system." in *Cambridge Workshop on Security Protocols*, 1995.

[6] T. P. Pedersen, "Electronic payments of small amounts," in *Proc. 4th International Security Protocols Conference*, 1996, pp. 59–68.
[7] R. L. Rivest and A. Shamir, "PayWord and MicroMint: Two simple micropayment schemes," in *Proc. 4th International Security Protocols Conference*, 1996, pp. 69–87.
[8] S. Foley, "Using trust management to support transferable hash-based micropayments," in *Financial Cryptography 2003*, 2003, pp. 1–14.
[9] S. Foley and T. B. Quillinan, "Using trust management to support micropayments," in *Annual Conference on Information Technology and Telecommunications*, Oct. 2002.
[10] D. Clarke, J. Elien, C. Ellison, M. Fredette, A. Morcos, and R. Rivest, "Certificate chain discovery in SPKI/SDSI," *Journal of Computer Security*, vol. 9, no. 4, pp. 285–322, Jan. 2001.
[11] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, "Simple Public Key Certificate," Internet Draft, July 2000.
[12] "JSDSI: A Java implementation of the SPKI/SDSI standard," `http://jsdsi.sourceforge.net`.
[13] G. Navarro, S. Robles, and J. Borrell, "Role-based access control for e-commerce sea-of-data applications," in *Information Security Conference 2002*, ser. Lecture Notes in Computer Science, vol. 2433. Springer Verlag, September/October 2002, pp. 102–116.
[14] "JADE, Java Agent DEvelopemnt Framework," `http://jade.tilab.com`, telecom Italia Lab.
[15] "Foundation for Intelligent Physical Agents," `http://www.fipa.org`, fIPA Specifications.
[16] J. Ametller, S. Robles, and J. Borrell, "Agent Migration over FIPA ACL Messages," in *Mobile Agents for Telecommunication Applications (MATA)*, ser. Lecture Notes in Computer Science, vol. 2881. Springer Verlag, October 2003, pp. 210–219.
[17] J. Ametller, S. Robles, and J. Ortega-Ruiz, "Self-protected mobile agents," in *3rd International Conference on Autonomous Agents and Multi Agents Systems*. ACM Press, 2004.

Survey of the Survivability of IT Systems

Pentti Tarvainen
VTT Technical Research Centre of Finland
P.O. Box 1100, FIN-90571 Oulu, Finland
email: pentti.tarvainen@vtt.fi

***Abstract*— Failure of IT systems often causes a major loss of service. Thus their dependability has become an important issue. Recent facets of the dependability of IT systems, such as reliability, availability, safety, security, confidentiality, integrity, and maintainability do not address the needs of IT systems because they do not include the notion of a degraded service as an explicit requirement. The concept termed survivability is a precise notion of the forms of services that are acceptable in a system, the circumstances under which each form is most useful, and the fraction of time that is acceptable in degraded services. In this paper survivability is discussed as a necessary new facet of dependability. The contribution of this paper is to give system architects the latest knowledge on survivability in order to help them develop survivable IT systems. Definitions of dependability and survivability are presented and discussed. In addition, the key properties and survivability requirements, and the role of fault tolerance in survivable systems are discussed. Furthermore, survivability implementation techniques and examples of survivability architectures are introduced and discussed. Finally, software architecture design and analyzing methods and frameworks relevant to survivability are discussed.**

***Index Terms*—Dependability, reliability, security, survivability**

I. Introduction

MODERN society faces a substantial, and generally acknowledged, risk of IT systems failure or compromise with potentially serious consequences. Sectors such as energy, financial services, telecommunications, healthcare, and defense are potential application areas of such systems [4], [5], [6], [8], [12]. Despite the best efforts of security professionals, no amount of system hardening can assure that a system that is connected to an unbounded network, such as the Internet, will not be vulnerable to attack. From point of view of a system architect's practical work, it is important to know what survivability really means and how it can be applied to IT systems. Survivability, and the survivability requirements, of an IT system must be taken into account at the system architecture design phase.

Survivability in IT systems is a relatively new research area. The precise definition of survivability is still being debated, with a number of definitions proposed. Most commonly, survivability is defined as "the ability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures or accident" [8], [9], [10], [11], [18], [19]. The term system is used in the broadest possible sense, including networks and large-scale systems of systems. The term mission refers to a set of abstract requirements or goals.

Survivability is a necessary new branch of dependability [1], [2], [3], [5], [8], [9], [10], [11], [18]. It addresses explicit requirements for restricted modes of operation that preserve critical essential services in adverse operational environments. A survivable system is one that satisfies its survivability specification of essential services and adverse environments. Essential services are defined as the functions of the system that must be maintained when the environment is hostile or failures or accidents are detected that threaten the system. The discipline of survivability can help ensure that IT systems can deliver essential services and maintain such essential properties as performance, security, reliability, availability and modifiability despite the presence of intrusions. Unlike the traditional security measures that require central control or administration, survivability is intended to address unbounded network environments.

Survivability requirements can vary substantially, depending on the scope of the system, the criticality and the consequences of failure and interruption of service. The definition and analysis of survivability requirements is a critical first step in achieving system survivability [8]. Survivability must address not only software function requirements but also requirements for software usage, development, operation and evolution.

Survivability architectures offer an approach to tolerating faults in which the continued service element of fault tolerance differs from normal service. By introducing this type of fault tolerance, progress can be made towards meeting the survivability goals of IT systems.

The rest of the paper is organized as follows: definitions of dependability and survivability are presented and discussed in Section II,. In Section III key properties and survivability requirements, and the role of fault tolerance in survivable systems are discussed. Techniques for implementing survivability are discussed in Section IV and examples of survivability architectures are introduced. Software architecture design and analyzing methods and frameworks that take survivability into account are introduced in Section V, and, Section VI contains the concluding remarks.

Manuscript received October 4, 2004. This research work was conducted in the Moose project under the ITEA cluster projects of the EUREKA network, and financially supported by Tekes (the National Technology Agency of Finland).

P. Tarvainen is with VTT Electronics, VTT, the Technical Research Centre of Finland, P.O. Box 1100, FIN-90571 Oulu, Finland, pentti.tarvainen@vtt.fi.

II. Definitions Related to Survivability

A definition of dependability and different definitions of survivability based on recent research work are discussed in this section; unofficial and common definitions of survivability are also introduced and discussed.

A. Definition of Dependability

Dependability is the system property that integrates such attributes as reliability, availability, safety, confidentiality, integrity, maintainability and survivability [32]. The dependability of a computing system is its ability to deliver a service that can justifiably be trusted. The service delivered by a system is the behavior of a system as it is perceived by its user(s). A user is another system, physical or human, that interacts with the former at the service interface. The function of a system is what the system is intended to do, and is described by the functional specification. The correct service is delivered when the service implements the system function.

B. Definitions of Survivability

Software quality is depicted in IEEE 1061 [30] and represents the degree to which the software possesses a desired combination of quality attributes. Another standard, ISO/IEC 9126-1 [31], defines the software quality model for external and internal quality. Based on this model, there are six categories of characteristics - functionality, reliability, usability, efficiency, maintainability and portability - which are further divided into sub-characteristics. Either of these standards does not define survivability.

A preliminary scoping of the general survivability problem was suggested by a 1993 report, "Survivable Computer-Communication Systems: The Problem and Working Group Recommendations" [25], written for the U.S. Army Research Laboratory (ARL). The report outlines a comprehensive multifunctional set of realistic computer-communication survivability requirements and makes related recommendations applicable to U.S. Army and defense systems [22].

The precise definition of survivability is still being debated, with a number of definitions proposed as described in Table I. The definitions in the table are listed in chronological order, based on the year the definition was published. Also the respective references are also denoted in the table. Based on Table I, survivability in respect of IT systems is a relatively new research area and the content of the definition of survivability depends on the domain.

In the context of software engineering, Deutsch [2] has offered the first definition shown in the table. This definition is not sufficient for all needs [18]. If it were applied to IT systems in this form, the user of the system could not be sure which functions had been selected as "essential functions" nor under what circumstances (i.e., after what damage) these functions would be provided.

The Institute for Telecommunications Services, a part of the U.S. Department of Commerce, has created an extensive glossary of telecommunications terms in Federal Standard 1037C [53]. The glossary contains a definition of survivability for telecommunications systems (the second definition in Table I). This definition seeks a framework for defining a service after some form of damage and relates closely to the goal of defining survivability for IT systems [1], [18]. Furthermore, this definition includes the notion of a degraded or different service and requires that it be defined.

Specifically on IT systems survivability, Ellison et al. [3] introduce the third definition of survivability shown in Table I. While this definition is a good beginning, it does not have the precision needed to permit a clear determination of whether a given system should be considered to be survivable [1], [18]. The first problem is that much is implied by the phrases "essential services", "attacks and failures" and "timely manner". If nothing further is defined, it is not possible for the architect of a system to determine whether a specific design is adequate to meet the needs of the user community. More importantly, if a phrase such as "essential service" is not precisely defined, it might be the case for any specific system that the determination of what constitutes an essential service is left to the system's developers rather than being defined carefully by application experts. A second problem with a definition of this form is that it provides no testable criterion for the term being defined. Essential services are defined as the functions of the system that must be maintained when the environment is hostile, or failures or accidents that threaten the system are detected.

Most commonly [8], [9], [10], [11], [18], [19], survivability

TABLE I
Definitions of Survivability

	Definition	Year	Domain	Ref.
1.	Survivability is the degree to which essential functions are still available even though some part of the system is down.	1988	IT systems in general	[2]
2.	Survivability is a property of a system, subsystem, equipment, process or procedure that provides a defined degree of assurance that the named entity will continue to function during and after a natural or man-made disturbance. Note: Survivability must be qualified by specifying the range of conditions over which the entity will survive the minimum acceptable level or post-disturbance functionality and the maximum acceptable outage duration.	1996	Telecommunication Systems	[54]
3.	Survivability is the ability of a network computing system to provide essential services in the presence of attacks and failures and recover full services in a timely manner.	1997	Network Computing Systems	[3]
4.	Survivability is the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures or accidents.	1999	Critical and Defense Systems	[8],[9], [10],[11], [18],[19]
5.	Survivability is the ability [of a system] to continue to provide service, possibly degraded or different, in a given operating environment when various events cause major damage to the system or its operating environment.	2000	Critical and Defense Systems	[1], [20]

is defined as the fourth definition in Table I. The term system, consisting of software and hardware, is used in the broadest possible sense, including networks and large-scale systems of systems. The term mission refers to a set of very high-level (i.e., abstract) requirements or goals. Missions are not limited to military settings since any successful organization or project must have a vision of its objectives, whether expressed implicitly or as an official mission statement. The terms attack, failure, and accident are meant to include all potentially damaging events, but these terms do not partition these events into mutually exclusive or even distinguishable sets. Attacks are potentially damaging events orchestrated by an intelligent adversary. Attacks include intrusions, probes and denials of service. Failures are included with accidents as part of survivability. Failures are potentially damaging events caused by deficiencies in the system or in an external element on which the system depends. Failures may be due to software design errors, hardware degradation, human errors or corrupted data. Accidents describe a broad range of randomly occurring and potentially damaging events, such as natural disasters. It is important to recognize that it is the mission fulfillment that must survive, not any particular subsystem or system component.

The fifth definition of survivability [1], [20] in Table I suggests a number of key points regarding the notion of survivability:

Survivability is a system property, relating the level of service provided to the level of damage present in the system and operating environment [7],

A system must be capable of providing different levels of service. In a system free of damage, the level of service should equate with full functionality. Different levels of service will correspond to varying subsets of functionality, where some functions that a system performs are obviously more critical than others [1], [18], and

The events that cause major damage can range from failures to attacks to accidents. It is often difficult to immediately determine the cause of the damage, e.g. whether the damage is the result of an intentional security attack or a random failure [3]. More important is the effect of the event in terms of damage to the system and operating environment — the amount of damage is central to the level of service that a survivable system can and should provide [7].

III. Requirements and Key Properties of Survivable Systems

In this section the key properties and requirements, and the role of fault tolerance in survivable systems are defined and discussed. The key properties of survivable systems are as follows [8]:

Firstly, central to the delivery of essential services is the ability of a system to maintain essential properties, i.e. specified levels of integrity, confidentiality, performance and other quality attributes. Thus it is important to define the minimum levels of quality attributes that must be associated with essential services.

Secondly, quality attributes are so important that definitions of survivability are often expressed in terms of maintaining a balance among multiple quality attributes, such as performance, security, reliability, availability, modifiability and affordability. Quality attributes represent broad categories of related requirements, so a quality attribute may contain other quality attributes. For example, the security attribute traditionally includes the three attributes of confidentiality, integrity and availability.

Thirdly, the ability to deliver essential services and maintain the associated essential properties must be sustained, even if a significant portion of the system is incapacitated. This ability should not be dependent upon the survival of a specific information resource, computation or communication link.

Fourthly, key to the concept of survivability is identifying the essential services, and the essential properties that support them, within an operational system. There are typically many services that can be temporarily suspended when a system is dealing with an attack or other extraordinary environmental condition. Such a suspension can help isolate areas affected by an intrusion and free system resources to deal with the intrusion's effects. The overall function of a system should adapt to the situation to preserve essential services. If an essential service is lost, it can be replaced by another service that supports mission fulfillment in a different but equivalent way. However, the identification and protection of essential services is an important part of a practical approach to building and analyzing survivable systems.

The survivability requirements of survivable systems can vary substantially, depending on the scope of the system, and the criticality and consequences of failure and interruption of service [8]. The definition and analysis of survivability requirements is a critical first step in achieving system survivability. Survivability must address not only the requirements for software functionality but also the requirements for software usage, development, operation and evolution. Five types of requirements definitions are relevant to survivable systems [8]: (1) System/Survivability Requirements, (2) Usage/Intrusion Requirements, (3) Development Requirements, (4) Operations Requirements and (5) Evolution Requirements.

Fault tolerance enables systems to continue to provide service in spite of the presence of faults. Fault tolerance consists of four phases: (1) error detection, (2) damage assessment, (3) state restoration and (4) continued service. Survivability is a dependability property; it is not synonymous with fault tolerance [1]. Fault tolerance is a mechanism that can be used to achieve certain dependability properties. In terms of dependability, it makes sense to refer to a system as reliable, available, secure, safe and survivable, or some combination, using the appropriate official definition(s). Describing a system as fault tolerant is really a statement about the system's design, not its dependability. While fault tolerance is a mechanism by which some facets of dependability might be achieved, it is not the only mechanism. Other techniques, such as fault avoidance, can also be used. In similar ways, fault elimination and fault forecasting can be used as mechanisms to improve a system's dependability.

IV. Implementation Techniques for Survivability

This section discusses implementation techniques related to the survivability of IT systems. In addition, examples of survivability architectures of IT systems are introduced and discussed

A. Survivability and Security

It is important to recognize the relationship between survivability and security. An application may employ security mechanisms, such as passwords and encryption, and may still be very fragile [17]. For instance, it may fail when a server or a network link dies. On the other hand, a survivable application must be able to survive some malicious attacks. Therefore, survivability must involve security. There are two aspects of survivability [17]: (1) survival by protection and (2) survival by adaptation.

Security mechanisms like access control and encryption attempt to ensure survivability by protecting applications from harmful, accidental or malicious changes in the environment [17]. The application could also survive by adapting itself to the changing conditions. These two aspects may not be mutually exclusive; as part of survival by adaptation, an application may utilize security mechanisms. For example, it may start using access control or increase the key length when it perceives the threat of an intrusion. Most current applications fail rather than adapt when QoS assumptions turn out to be too optimistic. The problem is made worse by the fact that survivability mechanisms are complicated, have little to do with an application's functionality, and developers only have limited tool support for incorporating them.

Furthermore, based on [17], the general notion of survival by adaptation results from years of experience in designing, implementing and deploying wide-area distributed systems, and is based on the ARMD (Adaptive, Redundant, Monitored, and Diversified) principles. These principles are not independent, and they need to be organized in a meaningful way to lead to survivability. For instance, being adaptive generally means that the adaptation is driven by some kind of monitoring. However, monitoring could be used without any kind of adaptation at all. Similarly, redundancy and diversity could very well be used without any adaptation, but in the context of adaptation they often define or broaden the scope of adaptation. Not all adaptive behaviors lead to survivability. For instance, shutting an application down on an exception indicating a change in the environment does not add anything to the survivability of the application. In fact, such an adaptation facilitates a whole class of denial of service attacks, whereas survivability is about continuing to provide useful service despite environmental changes.

Survival by adaptation typically involves monitoring and changing the Quality of Service (QoS) available to applications [17]. An application's design always makes some assumptions about the QoS provided by the environment for bandwidth, reliability, security services, etc. When these assumptions are violated, the application should try to adapt rather than fail. Most current applications, however, fail rather than adapt when QoS assumptions turn out to be too optimistic. The problem is made worse by the fact that survivability mechanisms are complicated, have little to do with an application's functionality, and developers only have limited tool support for incorporating them.

One important technique for improving system dependability and survivability is to provide mechanisms for a system to adapt at run time in order to accommodate varying resources, system errors and changing requirements [52]. For such self-repairing systems, one of the difficult problems is determining when a change is needed, and knowing what kind of adaptation is required. Based on [52], the challenge is to engineer things so that the system adapts appropriately at run time. There are two problems with this [52]. First, information must be got out of the running system. This can be done by employing low-level monitoring mechanisms that cover various aspects of the executing system. The second problem is to translate architectural repairs into actual system changes. This can be solved by writing table-driven translators that can interpret architectural repair operators in terms of the lower level system modifications.

B. Survivability Architectures

Survivability architecture is a system architecture that is designed specifically to deal with certain non-local faults [50]. A significant difficulty arises when the various concepts involved in survivability architectures have to be evaluated because experimentation with real systems is precluded. One approach to dealing with this problem is to use operational models that can be built and studied in the laboratory using a developed experimentation system [50].

1) Survivability Architecture: Block, Evade, React (SABER). Paper [33] proposes a survivability architecture called SABER. SABER blocks, evades and reacts to a variety of attacks by using several security and survivability mechanisms in an automated and coordinated fashion. SABER integrates several different technologies in an attempt to provide a unified framework for responding to the wide range of attacks malicious insiders and outsiders can launch. Most commercial responses to the diverse array of vulnerabilities have been to apply several discrete solutions [33]: (1) utilization of network-based firewalls [34], [35], (2) deployment of network-based and host-based intrusion detection systems [36], [42] and (3) manual installation and deployment of patches [37], [38].

At present, SABER is in the prototyping stages, with several interesting open research topics. It currently makes use of the following reaction and protection mechanisms [33]: (1) a network Denial-of-Service (DoS) resistant and Secure Overlay Services (SOS) architecture [39], (2) Manuscript intrusion and anomaly detection tools, [43], [44], placed within service contexts to detect malicious activity as well as stealthy "scans and probes", (3) a process migration system [40] that can be used to move a service to a new location that is not (currently) targeted by an attacker, (4) an automated software-patching system [41] that dynamically fixes certain classes of software-based attacks, such as buffer overflows, and (5) a high-level coordination and control infrastructure to correlate and coordinate the information and control flow.

2) Intrusion Tolerant Distributed Object System (ITDOS): Intrusion Tolerant Distributed Object System (ITDOS)

TABLE II
FEATURES OF SURVIVABILITY ARCHITECTURES

Feature	SABER	ITDOS	C4I
Security Mechanism	Integrates several security and survivability mechanisms	Symmetric Encryption Session Keys	"Plan-Based Survivability", Mobile IP and Ad Hoc network protocols for military use
Reaction/Protection Mechanism	Process Migration System, Network Denial-of-Service (DoS), Secure Overlay Services (SOS), An automated software-patching system	Distributed Object Middleware, CORBA	"Plan-Based Survivability"
Intrusion Detection	Network- and host-based intrusion detection, Anomaly-, registry- and file-based detection, Surveillance detection	Fault Tolerant Multicast Protocol + CORBA	"Plan-Based Survivability"
Domain	Network Systems	Heterogeneous Distributed Middleware Systems	Mobile Military Tactical Systems
Maturity	Prototype	Prototype	Prototype
Publishing Year	2003	2002	1999
Reference	[33]	[47]	[51]

architecture [47] discusses some of the challenging technical issues related to intrusion tolerance in heterogeneous middleware systems. The intrusion tolerant systems provide integrity and availability services in the face of successful attacks from an adversary.

Middleware is one area in which a system can provide intrusion tolerance [47]. Middleware is a very useful category of software that removes much of the tedium of distributed systems programming and shields programmers from having to deal with the numerous kinds of heterogeneity inherent in a distributed system [48]. Distributed object middleware is considered the most general kind of middleware, and CORBA [49] is a widely adopted standard for distributed object middleware. Middleware provides an ideal platform for intrusion tolerance extensions because it allows a variety of applications to be built that can transparently take advantage of the intrusion tolerance properties of the middleware, eliminating the need for custom solutions for each application [47].

3) Middleware Architecture for Secure and Survivable Mobile C4I Systems: An overview of a middleware-based mobile C4I (Command, Control, Communications, Computers, & Intelligence) architecture is discussed in [51]. The architecture is an outgrowth of work on a mobile distributed operating system that attempts to deal with various shortcomings in the Mobile Internet Protocol (Mobile IP) for military use. The architecture provides a foundation for balanced treatment of the complex, and frequently conflicting, dependability requirements (i.e. security, survivability, etc.) of military tactical systems.

The survivability architecture presented in [51] is controversial in that the Session Layer performs the primary communications function of a mobile "hand-off", instead of relying exclusively on the Network Layer to perform this function. This approach is defended on the basis that where tactical survivability is paramount, a mobile "hand-off" must be carefully controlled by the Session Layer, even if not specifically performed by that layer. The popular private sector approaches (e.g. Mobile IP) attempt to provide a "virtually stationary" environment by use of address mappings, which, except for performance impact, completely hide motion effects in the Network Layer. Such mobile networking approaches are unsuitable for military mobile C4I use, unless they are modified or designed to carefully coordinate their resource-management facilities with the survivability mechanisms of the Session Layer [51]. The mobile C4I architecture is a part of an evolving paradigm for C4I survivability called the Plan-Based Survivability, which seems to be able to solve many open problems with current survivability technology, and which has already been partly demonstrated by a working prototype. In effect, Plan-Based Survivability is a "superstructure" for unifying a diverse hierarchy of C4I defenses, both physical and informational.

C. Summary

Table II summarizes the features of the survivable architectures mentioned above. As a conclusion, the maturity of the architectures is insufficient for practical utilization in a system architect's work, but they will help to understand and solve the problem of survivable systems. The technical approaches of the architectures heavily depend on the system domain.

V. DESIGN OF SURVIVABILITY

In this section the available design and analysis methods and models, as well as frameworks relevant for the survivability of IT systems, are introduced and discussed.

A. Architecture Tradeoff Analysis Method (ATAM)

Paper [8] outlines an approach that addresses how to evaluate the ability of a system to deliver essential functions in an environment that includes intrusion scenarios. The general approach to survivability and security is consistent with the ATAM [14]. ATAM is a method for evaluating architecture-level designs that consider such multiple quality attributes as modifiability, performance, reliability and security to gain insight as to whether the fully fleshed out incarnation of the architecture will meet its requirements [14]. The method identifies tradeoff points between these attributes, facilitates communication between stakeholders (such as user, developer, customer, maintainer) from the perspective of each attribute, clarifies and refines the requirements, and provides a

framework for an ongoing, concurrent process of system design and analysis.

B. The Survivable Network Analysis Method (SNA)

A four-step SNA method [23] has been developed for analyzing survivability in distributed systems. SNA is a practical engineering process that enables systematic assessment of the survivability properties of proposed and existing systems, and modifications to existing systems. The analysis can be carried out at the lifecycle, requirements or architecture level. Based on [23], although the SNA method is developed for use with large-scale distributed-network systems, it is equally applicable to other architectures, including host-based and real-time systems. SNA's scenario-based approach is a generalization of the operation sequence and usage scenario methods.

C. The Willow Survivability Architecture

The Willow Architecture [13] is designed to enhance the survivability of IT systems and is a comprehensive approach to survivability in distributed applications. Based on [13], survivability is achieved in a deployed system using a unique combination of (1) fault avoidance by disabling vulnerable network elements intentionally when a threat is detected or predicted, (2) fault elimination by replacing system software elements when faults are discovered, and (3) fault tolerance by reconfiguring the system if non-maskable damage occurs.

D. Open Implementation Toolkit for Building Survivable Applications (QuO)

In [17] Pal, Loyall, Schertz and Zinky consider two aspects of survivability - namely, survival by adaptation and survival by protection. They show how the Quality Objects (QuO) distributed adaptive middleware framework enables the introduction of these aspects of survivability in a flexible and systematic manner. Furthermore, they also describe a toolkit for developing adaptive applications and demonstrate how more survivable applications can be built using the toolkit.

E. A Survivability Framework for Wireless Access Networks

Based on [21], a Survivability Framework for Wireless Access Networks consists of four layers, with survivability strategies possible in each layer. The four layers are termed access, access link level, transport and intelligent. The logical layers are independent of the physical implementation of the network. Each of the four layers is characterized by network functions, network components and communication links. This framework includes metrics for quantifying network survivability, possible survivability strategies, and restoration techniques for each layer.

F. An Architectural Framework and Algorithms for Engineering Dynamic Real-Time Distributed Systems

In [24] Ravindran presents a resource management architecture for engineering dynamic real-time, military, computer-based, Command and Control (C2) systems using commercial off-the-self technologies. In the proposed architecture a real-time system application is developed in a general-purpose programming language and system description language is used to specify the architectural-level description of the system. An abstract model that is constructed from the language specification is dynamically augmented by the System Description Language Runtime System to produce a dynamic Intermediate Representation (IR). The dynamic IR characterizes the state of the system and is used by a recourse management element to deliver the desired application QoS. The middleware techniques achieve the timeliness and survivability requirements through runtime monitoring and failure detection, diagnosis and dynamic recourse allocation.

G. Easel Modeling and Simulation Language

Easel [15] is a modeling and simulation programming language primarily intended for the research, analysis and depiction of unbounded systems, survivable architectures and emergent algorithms in applications, including Internet security, ad hoc communication networks, electric power and cooperation among autonomous vehicles. Easel is a notation

TABLE III
FEATURES OF THE SURVIVABILITY METHODS, MODELS, AND FRAMEWORKS

Method	Type of Method	Survivability Approach	Domain	Maturity Level	Ref.
ATAM	Design and Analysis Method	Intrusion Scenarios, Quality attributes	Not limited	High, widely used	[8], [14]
SNA	Design and Analysis Method	Scenarios	Large-Scale Distributed-Network Systems	High, based on ATAM	[23]
Willow	Modeling tool	Fault Avoidance, Fault Elimination, Fault Tolerance	Critical Distributed Applications	Medium	[13]
QuO	Modeling tool	Quality Objects, Adaptation, Protection	Middleware Applications	Medium	[17]
Survivability Framework for WANs	Modeling tool	Metrics, Restoration Techniques	Wireless Access Networks	Low	[21]
Architectural Framework	Modeling tool	System Description Language, Runtime Monitoring, Failure Detection, Dynamic Recourse Allocation	Military C2 Systems	High	[24]
Easel	Modeling and Simulation Language	Discrete Event Simulations Models	Unbounded Systems, Ad Hoc Networks	Medium	[15]

for describing abstract models of anything, a translator run-time system for running discrete event simulations from those models. An example of the use of Easel in network survivability analysis is presented in [16].

H. Summary

Table III summarizes the features of the survivability methods, models and frameworks described above. As a conclusion, the survivability approaches vary depending on the system domain. From the point of view of the system architect's practical work, there is a remarkable lack of suitable and mature methods. Only two (ATAM and SNA) design and analysis methods are available. The rest of the methods are modeling or simulation tools.

VI. Conclusion

Survivability is a new branch of dependability that addresses the explicit requirements for restricted modes of operation that preserve essential services in adverse operational environments. A survivable system is one that satisfies its survivability specification of essential services and adverse environments. In addition, survivability must address not only the requirements for software functionality but also the requirements for software usage, development, operation and evolution. Furthermore, survivability is a dependability property; it is not synonymous with fault tolerance. Fault tolerance is a mechanism that can be used to achieve certain dependability properties. In terms of dependability, it makes sense to refer to a system as reliable, available, secure, safe, and survivable, or some combination, using the appropriate definition(s). Describing a system as fault tolerant is really a statement about the system's design, not its dependability.

Survivability in respect of IT systems is a relatively new research area and the definition of survivability is still being debated. Two of the five definitions of survivability in Table I mention "essential services", and three of them mention the "degree of degraded or different" service to be provided by the survivable system, so these could represent points of agreement for a unified survivability definition. However, definition three mentions that full services will be recovered, whereas the other definitions only mention mission fulfillment, not full service recovery.

Security attacks are a major concern for IT systems. In some discussions survivability is viewed as synonymous with secure operation. A survivable application must be able to survive some malicious attacks, so survivability must involve security. There are at least two aspects of survivability: survival by protection and survival by adaptation. Security mechanisms like access control and encryption attempt to ensure survivability by protecting applications from harmful, accidental or malicious changes in the environment. Survival by adaptation typically involves monitoring and changing the QoS available to applications.

The maturity of the survivable architectures is insufficient for practical utilization in a system architect's work, but they will help to understand and solve the problem of survivable systems. Furthermore, there is a remarkable lack of suitable and mature methods, models and frameworks for practical use.

References

[1] J. C. Knight and K. J. Sullivan, "On the Definition of Survivability", University of Virginia, Department of Computer Science, Technical Report CS-TR-33-00, 2000.

[2] M. S. Deutsch and R. R. Willis, "Software Quality Engineering: A Total Technical and Management Approach", Englewood Cliffs, NJ: Prentice-Hall, 1988.

[3] R. J. Ellison, D. A. Fisher, R. C. Linger, H. F. Lipson, T. Longstaff and N. R. Mead, "Survivable Network Systems: An Emerging Discipline", Technical Report CMU/SEI-97-TR-013, Software Engineering Institute, Carnegie Mellon University, 1997.

[4] The Information Survivability Workshops of CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University. Available at: http://www.cert.org/research/isw.html, 13.02.2004.

[5] R. C. Linger and A. P. Moore, "Foundations for Survivable System Development: Service Traces, Intrusion Traces and Evaluation Models", Technical Report, CMU/SEI-2001-TR-029, Software Engineering Institute, Carnegie Mellon University, 2001. Available at: http://www.cert.org/archive/pdf/01tr029.pdf, 13.02.2004.

[6] K. Sullivan, J. Knight, X. Du and S. Geist, "Information Survivability Control Systems", Proceedings of the 21st International Conference on Software Engineering, Los Angeles, California, pp. 184-192, 1999.

[7] M. C. Elder, "Fault Tolerance in Critical Information Systems", Dissertation, Faculty of the School of Engineering and Applied Science, University of Virginia, 2001.

[8] R. J. Ellison, D. A. Fisher, R. C. Linger, H. F. Lipson, T. A. Longstaff and N. R. Mead, "An Approach to Survivable Systems", Technical Report, CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, 1999.

[9] I. Byon, "Survivability of the U.S. Electric Power Industry", Master's Thesis, Carnegie Mellon University, Information Networking Institute, 2000.

[10] J. Caldera, "Survivability Requirements for the U.S. Health Care Industry", Master's Thesis, Carnegie Mellon University, Information Networking Institute, 2000.

[11] R. J. Ellison, D. A. Fisher, R., C. Linger, H. F. Lipson, T. A. Longstaff and N. R. Mead "Survivability: Protecting Your Critical Systems", CERT Coordination Center Software Engineering Institute, IEEE Internet Computing, pp. 55-63, 1999.

[12] R. J. Ellison, L.C. Linger, T. Longstaff, and N. R. Mead "Survivable Network System Analysis: A Case Study", IEEE Software, Vol. 16, Issue: 4, pp. 70-77, 1999.

[13] J. Knight, D. Heimbigner, A. L. Wolf, A. Carzaniga, J. Hill, P. Devanbu and M. Gertz, "The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications", Technical Report CU-CS-926-01, Department of Computer Science, University of Colorado, Boulder, Colorado, 2001.

[14] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson and S., J. Carriere, "The Architecture Tradeoff Analysis Method", Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, IEEE Computer Society, 1998.

[15] WWW-pages of CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University. Available at: http://www.cert.org/easel/, 13.02.2004.

[16] A. M. Christie, "Network Survivability Analysis Using Easel", Technical Report, CMU/SEI-2002-TR-039, ESC-TR-2002-039, Software Engineering Institute, Carnegie Mellon University, 2002.

[17] P. P. Pal, J. P. Loyall, R. E. Schantz, J. A. Zinky and F. Webber, "Open implementation toolkit for building survivable applications", DARPA Information Survivability Conference and Exposition, Proceedings, Volume: 2, pp. 197 – 210, 2000.

[18] J. C. Knight, E. A. Strunk and K. J. Sullivan, "Towards a Rigorous Definition of Information System Survivability", DARPA Information Survivability Conference and Exposition, Washington DC, 2003.

[19] M. A. Hiltunen, R. D. Schlichting, C.A. Ugarte and G. T. Wong, "Survivability through Customization and Adaptability: The Cactus

Approach", DARPA Information Survivability Conference and Exposition, pp. 294-307, 2000.
[20] J. C. Knight, K. J. Sullivan, M. C. Elder and C. Wang, "Survivability Architectures: Issues and Approaches", DARPA Information Survivability Conference and Exposition, January 2000.
[21] D. Tipper, T. Dahlberg and H. Shin, "Providing Fault Tolerance in Wireless Access Networks", IEEE Communications Magazine, Vol. 40 Issue: 1, pp. 58-64, 2002.
[22] P. G. Neumann, "Practical Architectures for Survivable Systems and Networks", Technical Report supported by the U.S. Army Research Laboratory (ARL), Computer Science Laboratory, SRI International, 2000.
[23] N. R. Mead, R. J. Ellison, R. C. Linger, T. Longstaff and J. McHugh, "Survivable Network Analysis Method", Technical Report, CMU/SEI-2000-TR-013, ESC-2000-TR-013, Software Engineering Institute, Carnegie Mellon University, 2000.
[24] B. Ravindran, "Engineering Dynamic Real-Time Distributed Systems: Architecture, System Description Language, and Middleware", IEEE Transactions on Software Engineering, Vol. 28 Issue: 1, pp. 30-56, 2002.
[25] A. Barnes, A. Hollway and P. G. Neumann, "Survivable Computer-Communication Systems: The Problem and Working Group Recommendations", Technical report VAL-CE-TR-92-22 (revision 1), U.S. Army Research Laboratory, AMSRL-SL-E, White Sands Missile Range, NM 88002-5513, 1993.
[26] R. Kazman, G. Abowd, L. Bass and P. Clements, "Scenario-Based Analysis of Software Architecture", IEEE Software, Vol. 13, Issue: 6, pp. 47-55, 1996.
[27] M. Klein, T. Ralya, B. Pollak, R. Obenza and H. M. Gonzales "A Practitioner's Handbook for Real-Time Analysis", Boston, MA, Kluwer Academic, 1993.
[28] B. Boehm, "A Spiral Model of Software Development and Enhancement", ACM Software Eng. Notes 11, 4, pp. 22-42, 1986.
[29] M. Matinlassi, E. Niemelä and L. Dobrica, "Quality-driven architecture design and quality analysis method. A revolutionary initiation approach to product line architecture", VTT Electronics, Espoo, VTT Publications: 456, 2002. ISBN 951-38-5967-3, 951-38-5968-1
[30] IEEE Standard 1061- 1998, "Standard for a Software Quality Metrics Methodology", New York: The Institute of Electrical and Electronics Engineers, 1998.
[31] ISO/IEC 2001 - International Organization for Standardization and International Electrotechnical Commission. "Software engineering - Product quality - Part 1: Quality model". ISO/IEC 9126-1:2001(E)
[32] A. Avizienis, J.-C. Laprie and B. Randell, "Fundamental Concepts of Dependability", Technical report CS-TR-739, at University of Newcastle, 2001.
[33] A. D. Keromytis, J. Parekh, P. N. Gross, G. Kaiser, V. Misra, J. Nieh, D. Rubenstein and S. Stolfo, "A Holistic Approach to Service Survivability", Technical Report CUCS-021-03, Department of Computer Science, Columbia University, 2003.
[34] P. Thompson, "Web services – beyond HTTP tunneling". In W3C Workshop on Web Services, 2001.
[35] D. Moore, G. M. Voelker and S. Savage, "Inferring internet Denial-of-Service activity". In Proceedings of the 10th Usenix Security Symposium, pp. 9-22, 2001.
[36] D. Newman, J. Snyder and R. Thayer, "Crying wolf: False alarms hide attacks", Network World, June 2002. Available at: http://www.nwfusion.com/techinsider/2002/0624security1.html. 13.02.2004
[37] "Microsoft Security Tool Kit: Installing and Securing a New Windows 2000 System". Microsoft TechNet. Available at: http://www.microsoft.com/technet/security/tools/tools/w2knew.asp, 13.02.2004
[38] "RedHat 9 Security Advisories". Available at: https://rhn.redhat.com/errata/rh9-errata-security.html, 13.02.2004
[39] A. D. Keromytis, V. Misra and D. Rubenstein, "SOS: Secure Overlay Services". In Proceedings of ACM SIGCOMM, Pp. 61-72, 2002.
[40] S. Osman, D. Subhraveti, G. Su and J. Nieh, "The design and implementation of Zap: A system for migrating computing environments". In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI), pp. 361–376, 2002.
[41] S. Sidiroglou and A. D. Keromytis, "A Network Worm Vaccine Architecture". In Proceedings of the IEEE Workshop on Enterprise Technologies: Infrastructure for Collaborative Enterprises (WET-ICE), Workshop on Enterprise Security, 2003.
[42] M. V. Mahoney and P. K. Chan, "Learning non-stationary models of normal network traffic for detecting novel attacks". In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 376–385, ACM Press, 2002.
[43] D. L. Cook, W. G. Morein, A. D. Keromytis, V. Misra and D. Rubenstein, "WebSOS: Protecting Web Servers from DDoS Attacks". In Proceedings of the IEEE International Conference on Networks (ICON), 2003.
[44] S. Hershkop, R. Ferster, L. H. Bui, K. Wang, and S. J. Stolfo., "Host-based anomaly detection by wrapping file system accesses", Technical report, Columbia University, Department of Computer Science, 2003.
[45] P. Pal, M. Atighetchi, F. Webber, R. Schantz and C. Jones, "Reflections on Evaluating survivability: The APOD Experiments", The 2nd IEEE International Symposium on Network Computing and Applications (NCA-03), 2003.
[46] P. Pal, F. Webber, J. Zinky, R. Shapiro and J. Megquire, "Using QDL to specify QoS aware distributed (QuO) application configuration", IEEE Int'l Symp. Object-Oriented Real-Time Distributed Comp., 2000.
[47] D. Sames, B. Matt, B. Niebuhr, G. Tally, B. Whitmore and D. Bakken, "Developing a Heterogeneous Intrusion Tolerant CORBA System", International Conference on Dependable Systems and Networks (DSN'02), 2002.
[48] D. Bakken, "Middleware", Chapter in Encyclopedia of Distributed Computing, Urban, J., Dasgupta, P., eds., Kluwer Academic Publishers, 2001.
[49] "The Common Object Request Broker: Architecture and specification", OMG, Revision 2.5, 2001.
[50] J. C. Knight, K. J. Sullivan, M. C. Elder and C. Wang, "Survivability architectures: issues and approaches", DARPA Information Survivability Conference and Exposition, Proceedings, Volume: 2, pp. 157 -171, 2000.
[51] R. Browne, J. Valente and S. Hariri, "An advanced middleware architecture for secure and survivable mobile C4I systems", Military Communications Conference Proceedings, MILCOM 1999, IEEE, Volume: 1, pp. 506 -513, 1999.
[52] R. de Lemos, C. Gacek and A. Romanovsky, (Eds.), "Architecting Dependable Systems", Series: Lecture Notes in Computer Science, Vol. 2677, XII, 309 p., 2003, ISBN: 3-540-40727-8.
[53] "Federal Standard 1037C", U.S. Department of Commerce, National Telecommunications and Information Administration, Institute for Telecommunications Services, 1996. Available at: http://www.its.bldrdoc.gov/fs-1037/dir-001/_0065.htm, 09.03.2004.

Exploring Spyware Effects

Martin Boldt, Bengt Carlsson & Andreas Jacobsson
School of Engineering, Blekinge Institute of Technology, S-372 25 Ronneby, SWEDEN
{martin.boldt;bengt.carlsson;andreas.jacobsson}@bth.se

***Abstract* -- In this paper, we discuss various types of spyware programs, their behaviour, how they typically infect computers, and the propagation of new varieties of spyware programs. In two experiments, we investigate the occurrence and impact of spyware programs found in popular P2P applications. Based on the findings from the empirical investigations, we try to lift the perspective to a more general view on spyware deriving from the theory of (virtual) network effects. In a model, we categorize in what ways spyware might decrease the utility of belonging to a large virtual network. Here, the baseline is that spyware programs intrude systems and networks, but since they profit from user data they also intrude user privacy. In the model, the intrusions are classified as moderate, severe or disastrous. We found that spyware has the potential to overthrow the positive aspects of belonging to a large network, and network owners should therefore be very careful about permitting such programs in applications and on networks.**

***Index Terms* -- Malware, network effects, P2P, spyware.**

I. Introduction

During recent years, the world has seen the introduction of peer-to-peer (P2P) systems. P2P technology provides several beneficial solutions like, e.g., file-sharing, grid computing, web services, groupware and instant messaging (IM) [7]. P2P refers to a technology which enables two peers or more to collaborate in a network of equals [7] [10]. This may be done by using information and communication systems that are not depending on central coordination. P2P technology was first widely deployed and popularized by file-sharing applications such as KaZaa and IM tools like ICQ.

Even though there are several benefits with belonging to a large virtual network such as a P2P file-sharing network, the rising occurrence of malicious software (malware) may seriously impact the positive utility of using P2P applications. Usually, only the positive effects that increase utility are emphasized when discussing participation in large networks [5]. One example is the theory of virtual network[1] effects. Network effects are usually described as when the value of a product to one user depends on how many other users there are [11]. Often, utility of the system is proportional to the aggregate amount of resources that the participants are willing to put together. On information technologies, users generally benefit from utilising a popular format, system or application [11]. Typically, technologies subject to strong network effects tend to exhibit long lead times until a critical mass of users is obtained [5]. Then, explosive growth is followed. From the perspective of a network owner, a large network may help to create a strategic advantage useful for competition and growth purposes [1]. From the perspective of a network user, the larger the network is, the more valuable it will be to participants and users [1].

There are two kinds of feedback from network effects: positive and negative [11]. Positive feedback can be explained in that when a person joins a network, the network gets bigger and better, to everyone's benefit. However, large networks may also be exposed to negative feedback, which bring about significant risks and severe consequences for all of the network nodes. Therefore, negative feedback may decrease the utility of belonging to that network. To large networks, such as P2P file-sharing networks, there could be numerous examples of applications (e.g., malware), which contribute in creating negative effects that impact network utility. However, in this paper, we focus on one of these applications, namely spyware.

There are many different kinds of spyware, and hundreds of such programs exist throughout the Internet today [9]. Spyware programming is a relatively new computing phenomenon. Although there is no precise definition, the term "spyware" is typically used to refer to a category of software that, from a user's perspective, covertly gathers information about a computer's use and relays that information back to a third party. In this paper, we use the term spyware in conformity with this common usage. However,

[1] A virtual network describes a network of users bound together by a certain standard or technology, and where the exchange of information is the foundation for any information transaction. One example is the Internet.

in II, we look into and discuss some of the current views on the concept of spyware.

Even though most people are aware of spyware, it seems that the research community has spent limited effort on understanding the nature and extent of the spyware problem. However, so far there have been some initial research attempts (see for example [4] [9] [17]) of which this paper is an additional effort. On the other hand, most network practitioners and experts agree that spyware is a real problem with increasingly negative effects. One example of this view is derived from the Emerging Internet Threats Survey 2003 [3], which states that one in three companies have detected spyware on their systems, while 60% consider spyware to be a growing and future threat. Also, 70% of the companies consider that file-sharing over P2P networks is creating an open door into their organisation. Another example is an investigation made by Earthlink (one of the major American ISPs) [13]. Earthlink set to measure the occurrence of spyware on more than 2 million computers connected to their network. A total number of 12.1 million different spyware types were detected. Out of these, Trojan horses and system monitors approached 700 000 instances, and the remaining 11.4 million instances were classified as adware. Also, experts suggest that spyware infect up to 90% of all Internet-connected computers [13].

In summary, spyware is a problem that should be taken seriously, because it may have the potential to threaten the utility of belonging to a large virtual network. In this paper, we focus on exploring the effects of spyware programs that are bundled with several P2P applications. The aim is to investigate the implications on system capacity, network bandwidth, security and privacy. Besides introducing results from empirical investigations, we also discuss the network effects of spyware.

The paper is organised as follows. First, we give an introduction to spyware, in which we discuss the various kinds of spyware programs, their behaviour, how they typically infect computers, and the proliferation of new varieties of spyware. Next, we investigate the occurrence and impact of spyware programs found in popular P2P applications. In IV, we discuss the findings from the experiments and also try to lift the perspective to a more general view on spyware deriving from the theory of virtual network effects. In the end, conclusions are presented.

II. On Spyware

A. The Background of Spyware

As stated by [9], spyware exists because information has value. The idea with spyware is simply to fetch information. If a software developer can get revenue from advertisers, the owner can afford to make the software available for free. The developer is paid, and the user gets free, quality software. Usually, the developer provides two versions of the software, one for which the user has to pay a fee in order to receive, and one version that is freeware supported by advertising. In these cases, free software typically includes programs set to display advertisements and offers to the users (that is; adware). Therefore, the user can choose between the free software with the slight inconvenience of either pop-up ads or banners, or to pay for software free of advertising. So, users pay to use the software either with their money or with their time.

This method of including rather benign adware when developing and distributing free software was common until marketers noted three separate trends that pushed the development of adware into a different direction. The background was that:

--standard banner ads on the Internet were not delivering as well as expected (1% click-trough was considered good) [15],

--targeted Internet advertising typically performed much better [14], and

--while office hours were dead-time for traditional advertising (radio, TV, etc.), many analyses showed a surprisingly high degree of personal Internet usage during office hours [14].

The conclusion was that targeted Internet advertising was a whole new opportunity for the marketing of products and services. All that was required was a method for monitoring users' behaviour. So, once the adware was monitoring users' Internet usage and sending user details back to the advertiser, banners more suited to the users' preferences and personality was sent to the users in return. The addition of monitoring functionality turned adware into spyware, and the means to target advertising to interested parties accelerated [15]. In reality, the data collected by spyware is often sent back to the marketing company, resulting in display of specific advertisements, pop-up ads, and installing toolbars showed when users visit specific web sites. In this sense, spyware programs became technologies used to fetch valuable customer information.

B. The Operations of Spyware

The usual method for a spyware is to run secretly in the background of the users' computers [6]. The reason for this concealing of processes is commonly argued as that it would hardly be acceptable if, e.g., free file-sharing software kept stopping to ask the user if he or she was ready to fetch a new banner or a pop-up window [15]. Therefore, the client/server routine of spyware is normally executed in the background. In practice, there would be nothing wrong with spyware running in the background provided that the users know that it is happening, what data is being trans-

mitted, and that they have agreed to the process as part of the conditions for obtaining the freeware. However, most users are unaware of that they have software on their computers that tracks and reports on their Internet usage. Typically, a spyware program covertly gathers user information and spreads it without the user's knowledge of it. Once installed, the spyware monitors, e.g., user activity on the Internet and transmits that information in the background to third parties, such as advertising companies. In reality, spyware run constantly, even when their carrier program, e.g., a file-sharing tool, has been terminated.

A more or less legal grey area is exploited by the spyware actors, since they in most program licenses specify that information may be gathered for corporate purposes. However, the usual model is to collect more information than have been asked for [15]. Besides this, most license agreements are formulated in such a way that they are extensively hard for users to understand.

C. *The Types of Spyware*

There are many different kinds of spyware. For instance, one of the leading anti-spyware tools, PestPatrol, has a record of over 1400 instances of spyware published on their web site [8]. In order to make the spyware domain more graspable, we present the following classes of spyware. This classification is in conformity with a recently published study on measurement and analysis of spyware [9], although when presented here, the order of spyware types ranges from minimum to maximum user impact:

--Cookies and web bugs: Cookies are small pieces of state stored on individual clients' on behalf of web servers. Cookies can only be retrieved by the web site that initially stored them. However, because many sites use the same advertisement provider, these providers can potentially track the behaviour of users across many Internet sites. Web bugs are usually described as invisible images embedded on Internet pages used for locating a connection between an end user and a specific web site. They are related to cookies in that advertisement networks often make contracts with web sites to place such bugs on their pages. Cookies and web bugs are purely passive forms of spyware, they contain no code of their own. Instead they rely on existing web browser functions.

--Adware: Adware is a more benign form of spybot (see below). Adware is a category of software that displays advertisements tuned to the user's current activity. Although most "genuine" adware programs only display commercial content, some hybrids are involved in reporting the aggregate or anonymised user behaviour to a third party, as described in A.

--Tracks: A "track" is a generic name for information recorded by an operating system or application about actions that the user has performed. Examples of tracks include lists of recently visited web sites, web searches, web form input, lists of recently opened files, and programs maintained by operating systems. Although a track is typically not harmful on its own, tracks can be mined by malicious programs, and in the wrong context it can tell a great deal about a user.

--Browser hijackers: Hijackers attempt to change a user's Internet browser settings to modify their start page, search functionality, or other browser settings. Hijackers, which predominantly affect Windows operating systems, may use one of several mechanisms to achieve their goal: install a browser extension (called a "browser helper object"), modify Windows registry entries, or directly manipulate and/or replace browser preference files. Browser hijackers are also known to replace content on web sites with such promoted by the spyware authors [12].

--Spybots: Spybots are the prototypes of spyware. A spybot monitors a user's behaviour, collects logs of activity and transmits them to third parties. Examples of collected information include fields typed in web forms, lists of e-mail addresses to be harvested as spam targets, and lists of visited URLs. A spybot may be installed as a browser helper object, it may exist as a DLL on the host computer, or it may run as a separate program launched whenever the host operating system boots.

--System monitors: System monitors record various actions on computer systems. This ability makes them powerful administration tools for compiling system diagnostics. However, if misused system monitors become serious threats to user privacy. Keyloggers are a group of system monitors commonly involved in spyware activities. Keyloggers were originally designed to record all keystrokes of users in order to find passwords, credit card numbers, and other sensitive information.

--Malware: Malware is a set of instructions that run on a computer and make the system do something that an attacker wants it to do [12]. Malware refers to a variety of malicious software that includes viruses, worms, and Trojan horses. Spyware is one form of malware, but as will be discussed later on, spyware may also include instructions for downloading and installing, e.g., a virus.

Spyware succeeds because some of today's desktop operating systems make spyware simple to build and install [9]. Many instances of spyware have the ability to self-update, or automatically download new versions of themselves to the local host. Self-updating allows spyware authors to introduce new functions over time, but it may also be used to evade anti-spyware tools by avoiding specific signatures contained within the tools' signature databases using polymorphic techniques.

D. On the Implications of Spyware

Spyware may occupy resources of the computer that it infects or alter the functions of existing applications on the affected computer to the benefit of a third party. In that sense, spyware poses several risks. One commonly argued is that spyware compromises a user's privacy by transmitting information about that user's behaviour [4]. Even so, a spyware can also detract from the usability and stability of the computing environment of the user [9]. In addition, a spyware has the ability to introduce new security vulnerabilities to the infected host by downloading software updates [6]. Due to that spyware is widespread, such vulnerabilities put numerous amounts of computers at risk.

To summarize, the occurrence of spyware programs raise a real and growing threat to Internet usage in many aspects, and to other interested parties than only to end users. Four categories frequently argued on this topic are [3] [6] [15]:

--Consumption of system capacity: Spyware is often designed to be secretly loaded at system startup, and to partly run hidden in the background. Due to that it is not unusual for users to have many different instances of spyware running covertly simultaneously, the cumulative effect on the system's processing capacity can be dramatic.

--Consumption of bandwidth: The continual data traffic with gathering of new pop-ups and banner ads, and delivery of user data can have an imperative and costly effect on both private and corporate bandwidth.

--Security issues: Spyware covertly transmits user information back to the advertisement server, implying that since this is done in a covert manner, there is no way to be certain of exactly what data is being transmitted. Even though spyware, in its purest form, is a threat to privacy rather than security, some spyware programs have begun to act like Trojan horses. Most security experts would agree that the existence of spyware is incompatible with the concept of a secure system.

--Privacy issues: The fact that spyware operates with gathering and transmitting user information secretly in the background, and/or displays ads and commercial offers that the user did not by him-/herself chose to view, makes it highly privacy-invasive. Also, spyware enables for the spreading of e-mail addresses that may result in the receiving of unsolicited commercial e-mail (so called spam).

III. Experiments

We have developed a method for identifying and analysing spyware components and their behaviour on their host systems. This method has been used in several experiments (see, e.g., [4] [17]). In this section, we present the method applied in two experiments. Thereafter, a compilation of the experiment results is given.

A. Method

The method is tightly coupled with our security laboratory. Mainly because our experiment method is based on state preservation of computer systems, which can be provided due to the computer architecture of the security laboratory[2]. By storing the initial baseline state of a system it is later possible to conclude what changes occurred with regards to this baseline. In practice, this means that we store the state of a base system before installing any application carrying spyware components. Afterwards, it is possible to conclude any changes between the two. By also capturing all network data sent and binding that traffic to the corresponding program, we can correlate network data to specific programs. It is also possible to include measurements of, e.g., CPU and network utilization during the experiments.

By using this method, all systems that are measured consist of identical hardware and network setups. Therefore, operating systems and their applications are bitwise identical for all subjects in the experiment sample. This suffices for the generation of reliable results. In order to be sure that the results are derived from a certain spyware, we included a "clean" reference computer in the experiment.

Since file-sharing tools are notoriously known for bundling spyware, we used such applications in both of the experiments. In this context, it should be pointed out that no file-sharing activity took place in terms of sharing or downloading any content on the P2P networks. Our examination was limited to software versions released between January and May 2004, and as such, our observations and results might not hold for other versions. Also, we used an Internet surfing program that automatically simulated a user visiting 100 preconfigured Internet sites. This was an attempt to trigger any spyware to either leak this information to third parties or to hijack the web sessions. In order to identify and locate the spyware programs, several anti-spyware tools were used[3].

1) Experiment 1: In the first experiment, we investigated the occurrence and operations of five popular file-sharing tools[4]. More specifically, we examined spyware programs that were bundled with the file-sharing tools, the content and format of network data caused by spyware involved in Internet communication, and the extent of net-

[2] Throughout the experiments, we used 2.8Ghz Pentium 4 computers with 512MB primary memory.

[3] For a detailed list of the programs used, see http://www.ipd.bth.se/aja/SpywEffects_Ref.pdf

[4] The file-sharing tools were the standard (free) versions of BearShare, iMesh, KaZaa, LimeWire, and Morpheus.

work traffic generated by such programs. Even though there may be numerous components bundled with the installation of file-sharing tools, it was primarily the programs engaged in Internet communication that were of interest to us. There are two reasons for this. First, without this delimitation, the experiment data would be too comprehensive to grasp. Second, for spyware programs to leak user data, they must be involved in communication over the Internet.

2) Experiment 2: In the second experiment, we set to explore the effects in terms of resource usage that spyware bring about on a local system. A major problem introduced when setting up such an investigation involve how to choose the experiment sample. What we wanted was a program instance that was free of spyware and another instance (of the same program) that included spyware. Unfortunately it is almost impossible to remove only the spyware components and still have a working version of the original program since such components are very tightly coupled with the original program. We came to an acceptable solution by selecting KaZaa and KaZaa Lite K++ as the two subjects in the experiment sample. KaZaa Lite K++ is an instance of KaZaa where all spyware components have been removed by an independent group that reverse-engineered the original KaZaa program, carefully excluding or disabling all bundled components not solely used for file-sharing purposes. By using these two KaZaa versions, it was possible to subtract the resource utilization of KaZaa Lite K++ from the utilization of the original KaZaa and thereby receive a measurement of resources used by the spyware programs.

B. Results and Analysis

1) Experiment 1: A detailed list of the identified spyware programs is presented in TABLE I After having analysed the captured data, we concluded that all file-sharing tools contained spyware.

The two main carriers of spyware were iMesh and KaZaa (they included ten respectively eight programs each). The rates for the remaining file-sharing tools were five for Morpheus, four for LimeWire, and two for BearShare. In addition to these findings, we also discovered that all file-sharing tools contained spyware that were involved in Internet communication.

As can be seen in TABLE I, the retrieved spyware components were divided into “Adware” and “Spybot” based on their operations. We also included a category called “Download” because some of the components allowed for further software and/or updates to be downloaded and installed. In this category, examples such as hijackers and malware potentially could be included by the spyware distributors. In addition, all programs involved in any form of Internet communication were specified in a category called “Internet”. Finally, the category entitled “Host” specifies

TABLE I
IDENTIFIED SPYWARE PROGRAMS

Name	Host	Adware	Spybot	Download	Internet
BroadcastPC	M	x	x	x	**X**
KeenValue	K	x	x	**X**	**X**
Morpehus	M	**X**	x	**X**	**X**
BargainBuddy	I, K	x	x	x	
TopMoxie	L, M	x	x	x	
Cydoor	I, K	x	x		**X**
Gator	I, K	**X**	x		**X**
SaveNow	B	**X**	**X**		**X**
BonziBuddy	L	x	x		
Web3000	I	x	x		
ShopAtHomeSelect	I		**X**	**X**	**X**
WebHancer	K		x	x	
BrilliantDigital	K	x		**X**	**X**
MoneyMaker	L, M	**X**		**X**	**X**
Claria	I, K	x			**X**
iMesh	I	x			**X**
WeatherCast	B	x			**X**
CasinoOnNet	L	x			
MyBar	I, K, M	x			
New.Net	I			**X**	**X**
FavoriteMan	I			x	

which file-sharing tool that carried what spyware[5]. In the cases where our empirical results could confirm the view shared by anti-spyware tools, the markers in the table are declared with bolded capitol letters.

When analysing the outgoing network communication from the spyware components, we discovered that most of this traffic was not sent in clear text. This means that the transactions between the spyware components and their corresponding servers were either obfuscated or encrypted. This is also an explanation to why we were able to only identify two genuine spybot components. Since most traffic was sent in non-clear text, we could not really measure the extent to which such traffic was broadcasted. However, we did manage to identify some network traffic sent to spyware servers on the Internet that included, e.g., web sites visited, zip codes, country, and information about programs and operating system versions on the local host. In example, one of the spybot programs (ShopAtHomeSelect) that was found bundled with the iMesh file-sharing tool transmitted Internet browsing history records to several invoked servers on the Internet. The Internet records that were transmitted could be correlated to the web sites included in our preconfigured web surfing program.

2) Experiment 2: A compilation of the results from the resource utilization measurement can be seen in TABLE II

[5] B is for BearShare, I for iMesh, K is for KaZaa, L for LimeWire, and M for Morpheus.

TABLE II
RESOURCE UTILISATION MEASUREMENTS

	KaZaa Lite K++	KaZaa	Alteration
1. CPU usage (in%)	0.015	0.48	0.47
2. RAM usage (in%)	1.4	14	12.6
3. Addition of new files	50	780	730
4. Change in hard disk size (in MB)	8.6	46	37.4
5. Amount of network traffic (in MB)	0.6	29	28.4
6. No. of programs involved in Internet communication	1	11	10
7. No. of corresponding servers	60	349	289
8. No. of spyware programs installed	0	8	8

The measurements indicate that if KaZaa was installed, the rates for consumption of both system capacity (categories 1-4) and network bandwidth (categories 5-7) were significantly higher. This can be explained in that the spyware programs included in KaZaa affected both consumption of system capacity and network bandwidth. The high amount of network traffic was due to that the spyware components invoked numerous spyware servers on the Internet for the gathering of ads, pop-ups and banners. The accumulated local storage of collected commercial messages can have noticeable consequences on hard drive size, which also was the case for KaZaa.

In TABLE II, the measurements for the reference subject is subtracted from the file-sharing tools. The column entitled "Alteration" is represented by the difference between KaZaa and KaZaa Lite K++, that is; the spyware resource usage. Interestingly, three computer resources were significantly affected by the installation of spyware. In the first category of TABLE II, the occurrence of spyware had a measurable effect on CPU usage, KaZaa used 32 times more CPU capacity than KaZaa Lite K++. In category two, a significant difference was measured where the installation of KaZaa resulted in a ten times, or 65MB, increase of RAM usage. Finally, spyware programs had an imperative effect on the amount of network traffic generated by the file-sharing tools. More specifically, there was a 48 times augmentation of network traffic due to the spyware programs bundled with KaZaa. So, in contrast to KaZaa, installing a clean file-sharing tool (i.e., KaZaa Lite K++) caused marginal impact to system consumption and network bandwidth. However, due to the occurrence of spyware in file-sharing tools (see TABLE I), users with several such applications installed will, as a result of aggregate spyware activity, suffer from a continuos system and network degrading.

IV. DISCUSSION

Based on the findings in III, we can conclude that spyware programs exist, that they engage themselves in Internet communication, that they transmit user data, and that their existence have a negative impact on system and network capacity. Since we also can conclude that spyware programs are bundled with highly popular file-sharing tools[6], we can make out that spyware in accumulation may have a negative impact on networks and systems. In fact, the occurrence of spyware might decrease the overall utility of belonging to a large network such as a P2P file-sharing network. Thus, it might be relevant to elaborate on the theory of negative network effects to see whether spyware programs can threaten a large network.

In a model (TABLE III), we specify in what ways spyware might decrease the utility of belonging to a large virtual network. The baseline is that spyware programs intrude systems and networks, but since they profit from user data they also intrude user privacy. In the model, the intrusions are classified as moderate, severe and disastrous.

On user effects, some P2P providers include spyware in order to maximise profitability. Spyware may collect user data (such as e-mail addresses for spam distribution, surf records for personalised advertisement exposure, etc.) for commercial purposes. At present, spyware programs as such are rather benign, but cause problems to user privacy. In general, privacy is the right of individuals to control the collection and use of information about themselves [16]. This means that users should be able to decide for themselves, when, how, and to what extent information about them is communicated to others. Even though the user data exemplified in this category may not be that sensitive, spyware programs ignore user rights, and must therefore be considered privacy-invasive.

A more troublesome concern is the distribution of personal data, such as personal details (name, gender, hobby, etc.), e-mail conversation, and chat records. This may be the result of spyware techniques intended not only for commercial purposes, but also motivated by malicious intentions. Although, such spyware programs may not be that wide-spread today, a technological platform for these kinds of operations is available. This mean that although the probability of being infected by such a spyware is very low, the consequences may be devastating.

A third view would be if the spyware program updates on the servers were replaced with, e.g., keyloggers. In effect, harmful software could be distributed to vast groups of P2P tool users with the purpose of transmitting personally critical information such as financial data, private

[6] As an example, there are more than 350 million downloaded instances of KaZaa [2].

TABLE III
SPYWARE EFFECTS

	User	Computer	Network
Moderate	Commercially salable data	Consumption of capacity	Consumption of bandwidth
Severe	Personal data	Inferior code dissemination	Malware distribution
Disastrous	Critical data	Takeover	Breakdown

encryption keys, digital certificates or passwords. In reflection, financial threats from spyware programs may signify disastrous outcomes to vast groups of users.

In the experiments, we established a correlation between the presence of spyware programs and the consumption of computer capacity. Typically, spyware components utilised significant amounts of system resources, rendering in that computer resources were exploited in a larger extent than would otherwise be necessary. In accumulation, spyware operations degrade system capacity.

Also, it is problematic to comment on the quality of the code in the spyware programs, since the software requirements that have been used during the development process are left out in obscurity. The result can be that possibly inferior code is executed locally, which may have a negative influence on the entire system (i.e., not only to security). For example, as an effect of executing insufficient code, a system may lack performance or crash with, e.g., loss of important data as a result. In addition to this, software vulnerabilities may be exploited by malicious persons when breaking into a system, or when infecting it with destructive software (e.g., viruses).

As an utmost consequence, spyware programs deprive control over the system from the system owner. In effect, the installation of spyware programs may render in further installations of malware such as viruses and/or Trojans. Local services that are based on defect code and executed without the knowledge of the system owner are vulnerable to exploits, which may allow malicious actors to gain access over the computer. This is a disastrous situation because a takeover of system control affects both the local system and the surrounding network. A conquered system can be used as a platform for further distribution of malware.

At the network level, spyware operations in accumulation may contribute in network congestion. On one hand, the effects are unnecessary costs for network maintenance and expansion. On the other hand, network performance may be degraded. In either case, it is the network users that in the long run bear the costs.

The operations performed by spyware programs are approaching the operations of a virus with both a distribution and a payload part. Since users install, e.g., file-sharing tools that contain spyware programs on a voluntary basis, the distribution part is taken care of by the users themselves. This makes spyware programs function like a slowly moving virus without the typical distribution mechanisms usually otherwise included. The general method for a virus is to infect as many nodes as possible on the network in the shortest amount of time, so it can cause as much damage as conceivable before it gets caught by the anti-virus companies. Spyware, on the other hand, may operate in such a relatively low speed that it is difficult to detect. Therefore, the consequences may be just as dire as with a regular virus. The payload of a spyware is usually not to destroy or delete data, but to gather and transmit user information, which could be veritably sensitive. An additional complicating factor is that anti-virus companies do not generally define spyware as virus, since it does not typically include the ability to autonomously replicate itself. Overall, the nature of spyware substantiates the notion that malicious actions launched on computers and networks get more and more available, diversified and “intelligent”, rendering in that security is extensively problematic to uphold.

In theory, even a large network such as a P2P network may suffer an ultimate breakdown if it is continuously flooded with data. Should spyware programs continue to increase in number and to be more and more technologically refined, a network breakdown might be a final step. Although, in reality, this is not a plausible outcome. Nonetheless, if security and privacy risks are increasing as a result of being part of a P2P network, the positive value of using an application and thus belonging to that network will likely decrease. If users should experience that a threshold value (where the negative effects overthrow the positive aspects of using the application) is overstepped, then they will restrain from utilising that network. However, the experiment results indicate that even though spyware programs operate over P2P file-sharing networks, their effects are thus far rather modest. At least when it comes to system and network consumption. On the other hand, spyware programs that invade user privacy must be looked upon seriously. Spyware technologies mainly involved in gathering user data have a true value potential for marketers and advertisers. If these privacy-invasive activities should continue to evolve, there might be a great risk that spyware will be engaged in more malicious activities than simply fetching anonymised user/work station data. If so, that can lead to negative network effects and thereby cause a network to become less useful.

Hidden spyware components permit distribution of privacy-invasive information and security breaches within the network. Due to the construction of spyware, it may collect information that concerns other parties than only the work station user, e.g., telephone numbers and e-mail addresses to business contacts and friends stored on the desktop. In

the context that spyware usually is designed with the purpose of conveying commercial information to as many users as possible, not only the local user may be exposed to negative feedback of spyware. As well, the business contacts and friends may be the subjects of network contamination, e.g., receiving vast amounts of spam or other unsolicited content.

With the continuos escalation of spyware programs and the refinement of spyware technologies, network availability may be degraded to such an extent that ordinary transactions are overthrown by obscure malware traffic. A disastrous situation may occur where a network is seriously overloaded by malware distributed by computerised systems that are controlled by malicious actors. In conclusion, spyware activity may persuade users to abandon networks.

V. Conclusions

Based on the discussions of spyware and on the findings from the two experiments, we can conclude that spyware have a negative effect on computer security and user privacy. We have also found that a subsequent development of spyware technologies in combination with a continuos increase in spyware distribution will affect system and network capacity. A disastrous situation may occur if a network is seriously overloaded by different types of spyware distributed by computerised systems that are controlled by malicious actors. Then, the risk is a network breakdown. However, a more plausible outcome may be that users will abandon the network before that happens. In effect, spyware has the potential to overthrow the positive aspects of belonging to a large network, and network owners should therefore be very careful about permitting such programs in applications and on networks.

References

[1] S.-Y. Choi, D.O. Stahl, and A.B. Winston, "*The Economics of Electronic Commerce",* Macmillan Technical Publishing, Indianapolis IN, 1997.

[2] C|Net Download.com., http://www.download.com/, 2004-06-29.

[3] "Emerging Internet Threats Survey 2003", commissioned by Websense International Ltd., February, 2003. http://www.websense.com/company/news/research/Emerging_Threats_2003_EMEA-de.pdf, 2004-06-29.

[4] A. Jacobsson, M. Boldt, and B. Carlsson, "Privacy-Invasive Software in File-Sharing Tools", in *Proceedings of the 18th IFIP World Computer Congress,* Toulouse France, 2004.

[5] M.L. Katz, and C. Shapiro, "Systems Competition and Network Effects", in *Journal of Economic Perspectives 8:93-115*, 1994.

[6] M. McCardle, "How Spyware Fits into Defence in Depth", SANS Reading Room, SANS Institute, 2003. http://www.sans.org/rr/papers/index.php?id=905, 2004-06-29.

[7] A. Oram, "*Peer-To-Peer: Harnessing the Benefits of a Disruptive Technology"*, United States of America: O'Reilly & Associates Inc., 2001.

[8] PestPatrol, http://research.pestpatrol.com/Lists/NewPests(Pest Counts).asp, 2004-06-29.

[9] S. Sariou, S.D. Gribble, and H.M. Levy, "Measurement and Analysis of Spyware in a University Environment", in *Proceedings of the ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco CA, 2004.

[10] D. Schoder, and K. Fischbach, "Peer-to-Peer (P2P) Computing", in *Proceedings of the 36th IEEE Hawaii International Conference on System Sciences (HICSS'03)*, IEEE Computer Society Press, Los Alamitos CA, 2003.

[11] C. Shapiro, and H. Varian, *"Information Rules"*, HBS Press, Boston MA, 1999.

[12] E. Skoudis,*"Malware - Fighting Malicious Code"*, Prentice Hall PTR, Upper Saddle River NJ, 2004.

[13] "Spyaudit", commissioned by Earthlink Inc., http://www. earthlink.net/spyaudit/press/, 2004-06-29.

[14] J. Sterne, and A. Priore, *"E-Mail Marketing - Using E-Mail to Reach Your Target Audience and Build Customer Relationships"*, John Wiley & Sons Inc., New York NY, 2000.

[15] K. Townsend, "Spyware, Adware, and Peer-to-Peer Networks: The Hidden Threat to Corporate Security" (technical white paper), PestPatrol, 2003., http://www.pestpatrol.com/Whitepapers/PDFs/SpywareAdware P2P.pdf, 2004-06-29.

[16] A. Westin, "*Privacy and Freedom"*, Atheneum, New York NY, 1968.

[17] J. Wieslander, M. Boldt, and B. Carlsson, "Investigating Spyware on the Internet", in *Proceedings of the Seventh Nordic Workshop on Secure IT Systems*, Gjövik Norway, 2003.

Security properties for Intrusion Detection

Marco Giunti
Dipartimento di Informatica
Università Ca' Foscari di Venezia
giunti@dsi.unive.it

***Abstract*— We study security properties for real systems where classic access control policies are too strict; to prevent both direct and indirect accesses, our properties are then expressed as information flow properties. Particularly, we deal with mandatory integrity policies rather than secrecy policies. We argue these properties may help in discovering potential intrusions in a system, interpreting an intrusion as a sequence of actions which break the mandatory security policy of the system.**

***Index Terms*— Information flow, intrusion detection, access control, process algebras, formal methods, UNIX.**

I. Introduction

Intrusion Detection

The Intrusion Detection (ID) problem consists in discovering intrusions in a system and is typically related to the analysis of system logs, or *audit trail*. We immediately notice the absence, in the ID literature, of a formal and clear definition of "intrusion"; the generic idea is that an intrusion is something bad happened in the system, but no general definition for undesired behaviour is provided. For instance, in most Intrusion Detection Systems (IDS) (the tools which perform the log analysis), a non-root modification of the password file is considered an intrusion, but, for the best of our knowledge, there are no models explaining which general property is unsatisfied when overwriting the password file.

Starting from the 80's, when the Intrusion Detection problem was proposed [1], [2], two main ID approaches to audit trail analysis have been followed. Briefly, system logs analysis may follow an *anomaly detection* approach or a *misuse detection* approach. Anomaly based Intrusion Detection Systems (IDS), i.e. the tools that perform the analysis, try to detect intrusions statistically using metrics which define the "correct" behaviour of the system, considering any k-deviation from the correct metric as an intrusion, for some k [3], [4]. Misuse Intrusion Detection Systems, instead, check logs looking for a sequence of actions that match a system related attack signature; basically an attack signature is a sequence of steps which a malicious user may perform to compromise the security of the system, where a system may be local [5], [6] or distributed [7], [8]. Current work in the misuse detection area is most turned towards the formal definition of a language for signatures and the specification of algorithms for log auditing [9], [10], [11], [12].

Trying to formalize the intrusion detection problem, we note that none of these approaches is formal enough to tackle the ID problem. We do not want to rely on the anomaly detection approach for a formal definition of intrusion, since it's hard to demonstrate that statistically non-relevant attacks cannot be dangerous. For instance, the famous *sendmail* bug in BSD UNIX 4.2 [13], [5] gave rise to a public root shell after the execution by a standard user of three common used UNIX programs (cp, chmod and mail itself). We think that should be hard to discover these kind of attacks in a statistical approach, assuming no knowledge of the vulnerability. On the other hand, misuse approach security is based on the "perfect knowledge" of all possible attacks, that is, the definition of an intrusion set $\mathcal{I}$ is crucial. Then a system is secure if does not contain a log trace $t \in \mathcal{I}$. Here the main drawback we notice is that intrusions not yet known, or known in some different variant, cannot be discovered. Both these definitions of security appear thus to be unsound; in other words, stating a system is secure does not imply that the system does not contain intrusions.

A recent approach known as the *model-based analysis* [14], propose to analyze all the possible security-related behaviours of a system specified as an abstract model in a high level specification language. To express concurrency and non-determinism, a process algebra such Hoare's CSP [15] or Milner's CCS [16] may be used. The behaviours may be analyzed using automated verification techniques such as model checking. The authors of [14] propose to analyze all the states of the system for deciding if a bad event, for instance an overwriting of the password file, is happened in the system. This basically can be done by labeling as unsafe the states where the password file is overwritten and then by querying for the reachability of such states. More complex scenarios can be expressed by labeling entire execution sequences through temporal logical formulas.

An interesting point of this approach is that a complete log analysis is performed; that is, all possible executions are considered for detecting a possible intrusion. On the other hand, a general definition of security for the system is not provided: it is not clear to us which general property is unsatisfied when an unsafe state is reached.

As in [14], we analyze an abstract model of the system expressed in a process algebra; while we adopted a similar model, we moved from temporal logics based analysis to information flow properties study. Actually, we argue that intrusion detection is simply a special case of access control; given a mandatory security policy for the system, for instance the Biba policy [17], roughly synthesized as "no write up - no read down", then an intrusion is the sequence of steps which breaks the policy. Our approach consists in checking information flows of the system deciding through *non-interference*

properties if intrusions may happen in the system; the use of non-interference assure that both direct and indirect flows are checked. In this way, we don't assume *a priori* the knowledge of possible attacks; moreover, we can potentially find unknown attacks and generate patterns to be used as abstract signatures for misuse Intrusion Detection Systems.

A Non-Interference Property

In general, in a system, information is typically protected via some access control policy, limiting accesses of entities (such as users or processes) to data. There are different levels of flexibility of access control policies depending on the possibility for one entity to change the access rights of its own data. Here we will consider *mandatory* policies in which entities have no control on the access rights; historically [18] these strong mandatory security policies have been designed to avoid internal attacks performed by the so called *Trojan Horse* programs, i.e., malicious software that, once executed by a user, modifies the access rights of the data belonging to such a user. Unfortunately, even when direct access to data is forbidden by (strong) security policies, it might be the case that data are *indirectly* leaked by Trojan Horses which might exploit some observable system side-effects like, e.g., the CPU load or, more in general, the space/time availability of shared resources.

The necessity of controlling information flow as a whole (both direct and indirect) motivated Goguen and Meseguer in introducing the notion of *Non-interference* [19]. Non-Interference formalizes the absence of information flow within deterministic systems. Given a system in which *confidential* (i.e., high level) and *public* (i.e., low level) information may coexist, *non-interference* requires that confidential inputs never affect the outputs on the public interface of the system, i.e., never interfere with the low level users. If such a property holds, one can conclude that no information flow is ever possible from high to low level.

A possibilistic security property can be regarded as an extension of non-interference to non-deterministic systems. Starting from Sutherland [20], various such extensions have been proposed, e.g., [21], [22], [23]. All of these properties depend on the notion of behaviour that may possibly be observed of a system, i.e., the semantics model that is chosen to describe the system. Most of these properties are based on *traces*, i.e., the behaviour of systems that may possibly be observed is modeled through the set of their execution sequences.

In [24], Focardi and Gorrieri express the concept of non-interference in the *Security Process Algebra* (SPA) language, in terms of bisimulation semantics. In particular they introduce the notion of *Bisimulation-based non Deducibility on Compositions* (*BNDC*): a system E is *BNDC* if what a low level user sees of the system is not modified (in the sense of the bisimulation semantics) by composing any high level process Π with E. The main advantage of *BNDC* with respect to trace-based properties is that it is powerful enough to detect information flows due to the possibility, for a high level malicious process, to block or unblock a system. In particular, in [24], it is shown that a malicious process may build a channel from high to low, by suitably blocking and unblocking some system services accessible by low level users. The system used to build this covert channel turns out to be secure for trace-based properties. This motivates the use of more discriminating equivalences such as bisimulation.

In this paper we focus on integrity policies, i.e. policies which care about improper modification of data, rather than secrecy policies, i.e., policies which prevent unauthorized disclosure of data. In the literature, the standard integrity policy is known as the Biba model [17] and maybe roughly summarized through the rules "no read down", "no write up". The Biba policy is then the dual of the standard secrecy policy [18]; it follows that to express the Biba integrity policy as an non-interference property, we may say that a system is secure if no information can flow from low to high. By exchanging in the definition of *BNDC* the low level with the high level, we obtain a new property named *Bisimulation-based Non Modifiability on Compositions* (*BNMC* for short); we introduce this property in Section IV. *BNMC* thus can be used to detect whether a system satisfies the Biba policy, with respect to both direct and indirect accesses.

Unfortunately, the *BNMC* property does not help us too much in the analysis of real systems as UNIX-like operating systems or MS Windows OSs, since in these systems it is practically infeasible that no information flows from low to high. As a matter of fact in real systems, by default or by common practice, processes with root rights (that is, processes which have an high security level [18]) often have read access to low resources; in such case, expecting the system is *BNMC* is hopeless. On the other hand, we want to be able to distinguish the observable behaviour of systems where information flows from low to high, and thus we cannot use the *BNMC* property.

To this aim, we introduce a new security property that takes care of different types of accesses; in other words we may say that this information flow property takes into account different types of "modifiability", where for modifiability we intend a violation of the Biba's standard integrity policy.

We illustrate this new property with an example; we model a sample system into a CCS-like language named *read/write Security Process Algebra* (SPA$^{\mathsf{rw}}$, for short), a SPA [24] extension which we present in Section II. The set of visible actions of SPA$^{\mathsf{rw}}$ is partitioned in two levels as in [24], but, differently from SPA, each of these sets is in turn partitioned into read and write subsets; we indicate with $L^{\mathsf{r}}, L^{\mathsf{w}}$ the sets of low read and low write actions, respectively, such that $\overline{L^c} = L^c$ for each $c \in \{r, w\}$; analogously, we indicate with $H^{\mathsf{r}}, H^{\mathsf{w}}$ the sets containing high read and high write actions, respectively, such that $\overline{H^c} = H^c$ for both read and write capabilities. Moreover, $L = L^{\mathsf{r}} \cup L^{\mathsf{w}}$, i.e. the set L contains all low actions.

As an example, consider the process

$$P \stackrel{\triangle}{=} wannaread_{lw}.read_{hr}.0 \mid write_{hw}.0$$

where $wannaread_{lw} \in L^{\mathsf{w}}$, $read_{hr} \in H^{\mathsf{r}}$, $write_{hw} \in H^{\mathsf{w}}$. Basically, P waits for read and write requests but to read

an object it is before requested a confirmation consisting in pushing the $wannaread_{lw}$ button. We show that in P there is a flow from low to high; to this aim we define a low user

$$U \triangleq \overline{wannaread_{lw}}.0$$

The process U is low since in its definition only low level actions syntactically occur.

Informally, we may say that $P \in$ *BNMC* if the high observable behaviour of P does not change in the presence of any possible low level process interacting with P. We express the high observable behaviour of P in the terms of bisimulation semantics [16]: thus, we check if $(P \mid \Pi)\backslash_L \approx P\backslash_L$ for all low processes Π. Low actions are restricted (i.e. inaccessible from outside) since we want to observe the high behaviour of the system; this means that $P\backslash_L$ cannot do a move labelled with l, for any action $l \in L$. Intuitively, a binary relation $S \subseteq P \times P$ over processes is said to be a (weak) bisimulation if $(P, Q) \in S$ implies, for all moves α, that

1) Whenever $P \xrightarrow{\alpha} P'$ then for some Q', $Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q'$ with $(P', Q') \in S$
2) Whenever $Q \xrightarrow{\alpha} Q'$ then for some P', $P \stackrel{\widehat{\alpha}}{\Longrightarrow} P'$ with $(P', Q') \in S$.

P and Q are observation-equivalent or (weakly) bisimilar, written $P \approx Q$, if $(P, Q) \in S$ for some (weak) bisimulation S. Here $\Longrightarrow$ is a multi-step transition relation such that if $P(\xrightarrow{\tau})^* \xrightarrow{\alpha} (\xrightarrow{\tau})^* P'$ then $P \stackrel{\alpha}{\Longrightarrow} P'$; $\widehat{\alpha}$ is α if $\alpha \neq \tau$, otherwise α is ϵ, the empty sequence. We point out $P \stackrel{\alpha}{\Longrightarrow} P'$ implies $P \stackrel{\widehat{\alpha}}{\Longrightarrow} P'$. Here $P \notin$ *BNMC* as a high user may distinguish $(P \mid U)$ from P: that is, $(P \mid U)\backslash_L \not\approx P\backslash_L$. Particularly,

$$(P \mid U)\backslash_L \quad \xrightarrow{\tau} \equiv read_{hr}.0 \mid write_{hw}.0$$

and $(read_{hr}.0 \mid write_{hw}.0\ ,\ P\backslash_L)$ cannot[1] stay in any bisimulation since only the first process may do a $read_{hr}$ move. It follows that a low level user may interfere with the high activity of P inducing a high read move; since the observable action which determine $P \notin$ *BNMC* belongs to the H^{r} subset, we may informally say that the existing flow in P is from the low level to the high read level.

This policy is still too strict for real systems: even if in the system P above a flow from low to high is possible, this causes the modification of the high read behaviour in the system, rather than the high write behaviour. Since we deal with integrity, it seems reasonable to consider only flows which change the high write behaviour of systems.

To analyze the high integrity of the system, we consider a less restrictive property which, in analogy with the previous case, express the high write observable behaviour of the system. By ignoring high read actions in the definition of the *BNMC* property, we obtain a new security property named *Bisimulation-based non Write-Modifiability on Compositions* (*BNWMC*), which ensures that if a process satisfies this property, then it's high write observable behaviour is not modified by composing it in parallel with any possible low level process.

[1]To simplify examples, we will use often the well known congruence $P \mid 0 \equiv P$.

As one would expect, the process above satisfies *BNWMC*; intuitively, the reason is that:

$$\begin{array}{lll} (P \mid U)\,/_{H_r}\backslash_L & \xrightarrow{\tau} & \equiv \\ (read_{hr}.0 \mid write_{hw}.0)\,/_{H_r} & \approx & P\,/_{H_r}\backslash_L \end{array}$$

where the operator $/$ applied to H_{r} express the unobservability of high read actions, i.e. high read actions are relabelled to τ. Indeed, the action $read_{hr}$ is hidden in $(read_{hr}.0 \mid write_{hw}.0)\,/_{H_r}$ and thus we have that

$$(read_{hr}.0 \mid write_{hw}.0)\,/_{H_r} \xrightarrow{\tau} \equiv write_{hw}.0\ .$$

To match this move $P\,/_{H_r}\backslash_L$ does anything since $write_{hw}.0 \approx (wannaread_{lw}.read_{hr}.0 \mid write_{hw}.0)\,/_{H_r}\backslash_L$; this result follows from $a.P\backslash_a \approx 0$, for any process P and channel a.

The paper is structured as follows: in Section II we present some basic notions on the process algebra SPA$^{\mathsf{rw}}$; Section III contains the definition of a SPA$^{\mathsf{rw}}$ model for a small subset of a UNIX-based system. In Section IV we present the *BNWMC* property and we illustrate how this property can help in determining security of "real systems", analyzing the high write integrity of the Unix-like SPA$^{\mathsf{rw}}$ system model presented before. Comparison with related works and a few concluding remarks are reported in Section V.

II. Read/Write Security Process Algebra

In this section we present the syntax and the semantics of the *Read/Write Security Process Algebra* (SPA$^{\mathsf{rw}}$, for short) language.

The SPA$^{\mathsf{rw}}$ language is a variation of Milner's CCS [16], where the set of visible actions is partitioned into two levels, high and low, which are in turn partitioned into other two access type actions, read and write, in order to specify multi-level systems with read/write access control. SPA$^{\mathsf{rw}}$ syntax is based on the same elements as CCS that is: a set $\mathcal{L}$ of *visible* actions such that $\mathcal{L} = I \cup O$ where $I = \{a, b, \ldots\}$ is a set of *input* actions and $O = \{\bar{a}, \bar{b}, \ldots\}$ is a set of *output* actions; a special action τ which models internal computations, i.e., not visible outside the system; a complementation function $\bar{\cdot} : \mathcal{L} \rightarrow \mathcal{L}$ such that $\bar{\bar{a}} = a$, for all $a \in \mathcal{L}$. $Act = \mathcal{L} \cup \{\tau\}$ is the set of all *actions*.

The set of visible actions is partitioned into two sets, Act_H, Act_L, the high and low level action set, such that $\overline{Act_i} = Act_i$ for all $i \in \{H, L\}$. Moreover, each Act_i set, $i \in \{H, L\}$, is partitioned into two sets, $Act_i^{\mathsf{r}}, Act_i^{\mathsf{w}}$, respectively, of read and write actions, such that $\overline{Act_i^c} = Act_i^c$ for $c \in \{r, w\}$ and $i \in \{H, L\}$. Moreover, the sets Act_i are disjoint and they cover $\mathcal{L}$, i.e., $\bigcap_{i\in\{H,L\}} Act_i = \emptyset$ and $\bigcup_{i\in\{H,L\}} Act_i = \mathcal{L}$. For each $i \in \{H, L\}$, subsets Act_i^{r} and Act_i^{w} are disjoint and they cover Act_i: $\forall i \in \{H, L\} \quad \bigcap_{c\in\{r,w\}} Act_i^c = \emptyset$ and $\forall i \in \{H, L\} \quad \bigcup_{c\in\{r,w\}} Act_i^c = Act_i$.

The syntax of SPA$^{\mathsf{rw}}$ *terms* (or *processes*) is defined as follows:

$$E ::= \mathbf{0} \mid a.E \mid E + E \mid E|E \mid E \setminus v \mid E[f] \mid Z$$

Prefix	$\dfrac{-}{a.E \xrightarrow{a} E}$
Sum	$\dfrac{E_1 \xrightarrow{a} E_1'}{E_1 + E_2 \xrightarrow{a} E_1'} \quad \dfrac{E_2 \xrightarrow{a} E_2'}{E_1 + E_2 \xrightarrow{a} E_2'}$
Parallel	$\dfrac{E_1 \xrightarrow{a} E_1'}{E_1 \mid E_2 \xrightarrow{a} E_1' \mid E_2} \quad \dfrac{E_2 \xrightarrow{a} E_2'}{E_1 \mid E_2 \xrightarrow{a} E_1 \mid E_2'}$ $\dfrac{E_1 \xrightarrow{a} E_1' \;\; E_2 \xrightarrow{\bar{a}} E_2'}{E_1 \mid E_2 \xrightarrow{\tau} E_1' \mid E_2'} \;\; a \in \mathcal{L}$
Restriction	$\dfrac{E \xrightarrow{a} E'}{E \setminus v \xrightarrow{a} E' \setminus v}$ if $a \notin v$
Relabelling	$\dfrac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]}$
Constant	$\dfrac{E \xrightarrow{a} E'}{Z \xrightarrow{a} E'}$ if $Z \stackrel{\triangle}{=} E$

TABLE I
THE OPERATIONAL RULES FOR SPA^{rw}

where $a \in Act$, $v \subseteq \mathcal{L}$, $f : Act \to Act$ is such that $f(\bar{\alpha}) = \overline{f(\alpha)}$, $f(\tau) = \tau$, $f(Act_i^c) \subseteq Act_i^c \cup \{\tau\}$ for each $c \in \{r, w\}$ and $i \in \{H, L\}$, and Z is a constant that must be associated with a definition $Z \stackrel{\triangle}{=} E$.

Intuitively, $\mathbf{0}$ is the empty process that does nothing; $a.E$ is a process that can perform an action a and then behaves as E; $E_1 + E_2$ represents the nondeterministic choice between the two processes E_1 and E_2; $E_1|E_2$ is the parallel composition of E_1 and E_2, where executions are interleaved, possibly synchronized on complementary input/output actions, producing an internal action τ; $E \setminus v$ is a process E prevented from performing actions in v; $E[f]$ is the process E whose actions are renamed *via* the relabelling function f. For the definition of security properties it is also useful the *hiding* operator, /, of CSP [15] which can be defined as a relabelling as follows: for a given set $v \subseteq \mathcal{L}$, $E/v \stackrel{\text{def}}{=} E[f_v]$ where $f_v(x) = x$ if $x \notin v$ and $f_v(x) = \tau$ if $x \in v$. In practice, E/v turns all actions in v into internal τ' s. For the sake of brevity we will write often $P\backslash_L$ to indicate the process $P\backslash_{Act_L}$ and $P\,/_{H_r}$ to indicate the process $P/_{Act_H^r}$. We denote by $\mathcal{E}$ the set of all SPA^{rw} processes.

The operational semantics of SPA^{rw} processes is given in terms of *Labelled Transition Systems* (LTS, for short). A LTS is a triple $(S, A, \rightarrow)$ where S is a set of states, A is a set of labels (actions), $\rightarrow \subseteq S \times A \times S$ is a set of labelled transitions. The notation $(S_1, a, S_2) \in \rightarrow$ (or equivalently $S_1 \xrightarrow{a} S_2$) means that the system can move from the state S_1 to the state S_2 through the action a. The operational semantics of SPA^{rw} is the LTS $(\mathcal{E}, Act, \rightarrow)$, where the states are the terms of the algebra and the transition relation $\rightarrow \subseteq \mathcal{E} \times Act \times \mathcal{E}$ is defined by structural induction as the least relation generated by the inference rules reported in Table I.

Value-Passing SPA^{rw}: We briefly introduce a *value-passing* version of SPA^{rw}. The syntax is reported in Table II; value

$a(x_1, ..., x_n).E$, $\overline{a}(e_1, ..., e_n).E$, $\tau.E$	$Prefixes$
$\sum_{i \in I} E_i$	Sum
$E_1 \mid E_2$	$ParallelComposition$
$E \backslash v$	$Restriction\ (v \subseteq \mathcal{L})$
$E[f]$	$Relabelling$
if b **then** E	$Conditional$
if b **then** E **else** E	$Conditional$
$A(e_1, ..., e_n)$	$Costant$

TABLE II
VALUE-PASSING SPA^{rw} SYNTAX

expressions $e_1, ..., e_n$ need to be consistent with ariety of actions a and of constants A, respectively, whereas b is a boolean expression. A constant A is defined by $A(x_1, ..., x_m) \stackrel{def}{=} E$ where E is a value-passing SPA^{rw} agent that must not contain free variables except $x_1, ..., x_m$, which need to be distinct. As described in [16], value-passing calculus semantics is given by translating the calculus into the pure calculus: the idea is to translate each possible combination of the values into different SPA^{rw} actions and different SPA^{rw} processes.

III. THE SPA^{rw} MODEL FOR A SIMPLE UNIX-BASED SYSTEM

We present a SPA^{rw} model of a small subset of a UNIX-based system which will exhibit a vulnerability due to /etc/utmp, a file containing the association between logged users and related terminals. The following is quoted from the CERT Advisory CA-1994-06 (writable /etc/utmp vulnerability): "Description: If the file /etc/utmp is writable by users other than root, programs that trust the information stored in that file can be subverted. Impact: This vulnerability allows anyone with access to a user account to gain root access. Solution: The solutions to this vulnerability are to either (a) protect the file, or (b) patch all the programs that trust it."

As program which trust /etc/utmp, we consider *comsat*, a server which periodically checks the mailbox of any user for incoming mail and prints the first lines of the new message on the terminal in which the user is logged on. As mentioned above the terminal is contained in the file /etc/utmp. If this file is world-writable, a malicious user could substitute the /etc/passwd in the place of the terminal in which he/she is logged on and then send a mail to himself containing the line "root::0:0:". Comsat will overwrite the password file /etc/passwd with this message setting root's password to blank. The user can now login as root without providing a password.

The UNIX subset we consider consists in a simplified view of the file system and the comsat server itself. Our model represents an adapted SPA^{rw} specification of the model presented in [14].

We model the system within the value-passing version of the SPA^{rw} language (a pure calculus traduction is available in [25]). The values we pass consist in contents, levels, files and the system state itself; we will use $Cont, Lev, Files$ to indicate, respectively, the finite sets of contents, levels and files. For simplicity, since this is not relevant for the security analysis of the system, we consider a single mailbox, owned

by a standard (low) user, and a single terminal; thus "mailbox" should be a shorthand for the file /var/spool/mail/lowuser while "tty" should indicate /dev/tty.

$$\begin{array}{ll} Cont & = \{\, /etc/passwd, root :: 0 : 0 : \} \\ Files & = \{\, /etc/passwd, /etc/utmp, tty, mailbox \,\} \\ Lev & = \{\, L, H \,\} \end{array}$$

Each action of the value-passing calculus, apart τ, contains a security level $l \in Lev$; intuitively, value-passing channels with a parameter level l will be automatically translated in pure SPA$^{\mathsf{rw}}$ actions belong to Act_l. All value-passing actions (apart τ) are a priori classified as read or write; the translation of a channel classified as read (write) generates actions belonging to the subsets Act_i^{r} (Act_i^{w}), for $i \in \{H, L\}$.

We first model the file system $Fs(s)$ process as reported in Table III.

$$\begin{array}{ll} Fs(s) & \triangleq Write(s) + Read(s) \\ Write(s) & \triangleq write(f,l,c).Check(s,f,l,c) \\ Check(s,f,l,c) & \triangleq if \quad w \subseteq ACL(f,l) \\ & \qquad then\ Fs(update(s,f,c)) \\ & \quad else\ Fs(s) \\ Read(s) & \triangleq read(f,l).\overline{channel}(f,l,extract(f)).Fs(s) \end{array}$$

TABLE III
FILE SYSTEM

The file system process receives read and write requests; each request contains the security level of the process which invoke it both with the name of the file interested (plus the content, in the case of a write request): that is, $f \in Files, l \in Lev, c \in Cont$. Channel value-passing terms belong to read subsets; read and write value-passing terms belong to the obvious subsets.

In the file system, we implement the access control security policy of the UNIX subset we are considering. Roughly speaking, this is expressible as "no write-up", that is, no controls on the read accesses are needed to enforce this policy while low subjects cannot write high objects. In the value-passing version of the system we enforce this policy in such way. We statically define an access control matrix $ACL := Files \times Lev$ such that each entry of the matrix may contain r, w or both; that is, if $ACL(f,l)$ contains r, w then level l has the capability to read and write file f. We build the ACL matrix in such way. Following [18], we assign a security level to each object; that is, for each file $f \in Files$, we set a level $l \in Lev$ using a level function $level : Files \longrightarrow Lev$, $level(f) = l$.
Then we release capabilities w.r.t. the "no write up" policy:

$$\begin{array}{lll} \forall (f,l) \in ACL & l \geq level(f) \Longrightarrow & w \subseteq ACL(f,l) \\ \forall (f,l) \in ACL & & r \subseteq ACL(f,l) \end{array}$$

The access control list is defined in Table IV. We set $level(/etc/passwd) = high$, $level(tty) = high$, $level(/etc/utmp) = low$ and $level(mailbox) = low$; that is, */etc/passwd* and *tty* are the high objects of our system. According to the UNIX-like system scenario we are considering, processes with low level may read and write both "their" mailbox and the terminal file /etc/utmp, while they have only read access to /etc/passwd and tty; high processes can do everything.

	$/etc/utmp$	$mailbox$	$/etc/passwd$	$/dev/tty$
L	r, w	r, w	r	r
H	r, w	r, w	r, w	r, w

TABLE IV
THE ACCESS CONTROL LIST (ACL)

In the term $Fs(s)$, the variable s denotes the file system physical structure updated by the file system itself and represents the actual state of the file system. We model the file system as a Deterministic Finite Automaton; each state depicts a unique combination of values for the considered set of files, since both the set of values and the set of files are finite. Updating the file system structure consists thus in changing DFA's state; the automaton is deterministic since only the write requests lead to a state transition. It follows that, for a given state s, we are able to return the content, or value, of each file f. Action $write(f,l,c)$ inside agent $Write(s)$ represents the fact that the file system is waiting for a request from a security level l to write the content c on file f. Once the request arrived the agent $Check(f,l,c)$ establishes if level l may write file f: as said before, this is admissible if $w \subseteq ACL(f,l)$. If this holds, then we update the file system's structure s with the new content of f, which is c. Similarly, in agent $Read(s)$, once arrived a l-level request to read file f, then the content c of file f, which is returned by the function $extract : Files \rightarrow Cont$ in $extract(f)$, is sent through the channel $channel(f,l,extract(f))$. Here we do not change the automata state, since we are not updating the file system.

We briefly sketch the translation in the pure SPA$^{\mathsf{rw}}$ calculus of the value-passing term

$$Write(s) \triangleq write(f,l,c).Check(s,f,l,c)$$

for two files, $/etc/passwd$ and $/etc/utmp$ and one content, *root::0:0:*. We are thus considering two sets, namely $Files'$ and $Cont'$, respectively subsets of $Files$ and $Cont$: $Files' = \{/etc/pwd, /etc/utmp\}$, $C' = \{root :: 0 : 0 :\}$.

We translate the value-passing agents according to the access rights defined on the ACL matrix; parameters *pwd*, *utmp* and *r0* abbreviate respectively */etc/pwd*, */etc/utmp* and *root::0:0:*. Let

$$\begin{array}{ll} [[Fs(s)]]_{ACL} & = [[Write(s)]]_{ACL} + [[Read(s)]]_{ACL} \\ [[Fs(s)]]_{ACL} & \triangleq F_s \\ [[Fs(s1)]]_{ACL} & \triangleq F_{s1} \\ [[Fs(s2)]]_{ACL} & \triangleq F_{s2} \end{array}$$

Intuitively, s is the initial state of the file system, $s1$ is the state reached from s overwriting the value of */etc/passwd* with *root::0:0:* and $s2$ is the state reached from s overwriting the value of */etc/utmp* with *root::0:0:* . Thus the translation of the value-passing term $Write(s)$ is:

$$[[Write(s)]]_{ACL} \triangleq \begin{array}{ll} r0\_pwd_{lw} \,.\, F_s & + \\ r0\_pwd_{hw} \,.\, F_{s1} & + \\ r0\_utmp_{lw} \,.\, F_{s2} & + \\ r0\_utmp_{hw} \,.\, F_{s2} & \end{array}$$

As you may see, each write move implies an updating of the system except the low request to write the file $/etc/pwd$, since $low \not\geq level(/etc/passwd)$.

The translation of the value-passing agent $Write(s)$ in the pure calculus SPA$^{\mathsf{rw}}$ gives us the chance to show the intuition of how non-interference may help in the individuation of information flows within a system. Consider the process $[[Write(s)]]_{ACL}$, in particular the branch $r0\_utmp_{lw} \,.\, F_{s2}$. If the low process $\overline{r0\_utmp}_{lw}.0$ is composed in parallel with $[[Write(s)]]_{ACL}$, then the file system may evolve to $s2$ since

$$(\ [[Write(s)]]_{ACL} \mid \overline{r0\_utmp}_{lw}.0\ )\backslash_L \xrightarrow{\tau} \equiv F_{s2}\backslash_L$$

Low actions are restricted since we want to observe the high behaviour of the system. Now if $F_{s2}\backslash_L$ can do an α_h move, $\alpha_h \in Act_H$, but $[[Write(s)]]_{ACL}\backslash_L \stackrel{\widehat{\alpha_h}}{\not\Longrightarrow}$, then this represents a flow from low to high since now the high behaviour of the system is changed.

As a matter of fact, the core of the CERT Advisory CA-1994-06 (writable /etc/utmp vulnerability) consists in this flow. Moving over, we argue this flow may compromise the integrity of the system only if some new high *write* action is observable; that is, only if the unmatchable action is some α_{hw}, $\alpha_{hw} \in Act^{\mathsf{w}}_H$.

We present the specification of the comsat server in Table V. We let $Comsat$ actions belong to H, since, in the scenario we are considering, comsat process has *root* rights. *Comsat*, through the action $\overline{read}(mailbox, H)$, asks the file system to read the mailbox; then, the content of the mailbox, which for simplicity should be the last unread plain text message, will be stored in c_{msg} through action $channel(mailbox, H, c_{msg})$. If there is a message to print, then *Comsat* asks to read the /etc/utmp file to know on which tty the user (owner of the mailbox) is logged in; otherwise the mailbox does not contain new messages and the program goes back to the starting point. Again, for the sake of simplicity, we let the content c_{utmp} of /etc/utmp to consist in a file path associated to the unique mailbox of the system; that is, $c_{utmp} \in Files$, possibly $c_{utmp} = /dev/tty$. Then, *Comsat* forwards to the file system a request to write the new message c_{msg} on the path contained in c_{utmp}, which should be the tty terminal file where the mailbox owner is logged.

Finally the whole *System* we consider is obtained by the parallel composition of the agents introduced, where s is the init state of the file system, i.e the state where $/etc/utmp$ contains a link to $/dev/tty$, $/etc/passwd$ has value $secret$, the $mailbox$ is empty, and $/dev/tty$ is empty.

$$System \triangleq Fs(s) \mid Comsat$$

$$\begin{array}{rcl} Comsat & \triangleq & \overline{read}(mailbox, H).channel(mailbox, H, c_{msg}). \\ & & if\ c_{msg} \neq null\ then \\ & & \quad \overline{read}(utmp, H).channel(utmp, H, c_{utmp}). \\ & & \quad \overline{write}(c_{utmp},\ H,\ c_{msg}). \\ & & \quad Comsat \\ & & else\ Comsat \end{array}$$

TABLE V
COMSAT

IV. SECURITY PROPERTIES

The *BNDC* [24] security property aims at guaranteeing that no information flow from the high to the low level is possible, even in the presence of malicious processes. The main motivation is to protect a system also from internal attacks, which could be performed by the so called *Trojan Horse* programs, i.e., programs that are apparently honest but hide inside some malicious code.

We introduce here the converse security property *Bisimulation based Non Modifiability on Compositions (BNMC* for short), which denies information flows from low to high. Property *BNMC* is based on the idea of checking the system against all low level potential interactions, representing every possible low level malicious program. In particular, a system E is *BNMC* if for every low level process Π a high level user cannot distinguish E from $(E|\Pi)$, i.e., if Π cannot interfere with the high level execution of the system E. In other words, a system E is *BNMC* if what a high level user sees of the system is not modified by composing any low level process Π to E. *BNMC* thus may be used to verify if a system respects the Biba policy [17], w.r.t. both direct and indirect information flows. We indicate with $\mathcal{E}_L$ the set of *low* processes. More formally, let $\mathcal{L}(E)$ denote the *sort* of E, i.e. the set of the (possibly executable) actions occurring syntactically in E, $E \in \mathcal{E}$; then the set of low level agents is defined as $\mathcal{E}_L \triangleq \{E \in \mathcal{E} : \mathcal{L}(E) \subseteq Act_L \cup \{\tau\}\}$.

Definition 4.1 (BNMC): Let $E \in \mathcal{E}$.

$$E \in BNMC \quad \text{iff} \quad \forall\ \Pi \in \mathcal{E}_L,\ E\backslash_L \approx (E|\Pi)\backslash_L.$$

Example 4.2: $System \notin BNMC$. Intuitively this holds since comsat, to notify messages, carry out read accesses to the mailbox, which is writable by low users; it follows that the high observable behaviour of the whole system is influenced by the interaction with low processes.

To see that, it's enough to consider a low processes $U \triangleq \overline{write}(mailbox, L, r0).0.$ which sends a single message containing the line "root::0:0:" to himself. The process $(System \mid U)\backslash_L$ after two τ reductions exhibits the high action $\overline{channel}(mailbox, H, r0)$, while this action is unobservable after any number of τ reductions of $System\backslash_L$, simply because no one sended a mail with content *root::0:0:*.

This example shows the inadequacy, in such cases, of the *BNMC* security property. Actually, in Example 4.2, the high move which determine $System \notin BNMC$ consists in reading the content of the mailbox; that is,

$channel(mailbox, H, r0) \in Act'_H$. As a matter of fact, we observe that *BNMC* is too strong for the security analysis of real systems and thus do not help very much. For instance, if comsat specification was consisting only in checking the content of the mailbox, this system would be also considered unsafe.

To capture this situations that are very likely in real systems, we introduce a security property weaker than *BNMC* called *Bisimulation-based non Write-Modifiability on Compositions* (*BNWMC* for short). A system E is *BNWMC* if for every low level process Π a high level user cannot distinguish the *write* behaviour of E from the one of $(E|\Pi)$, i.e., if Π cannot interfere with the high level write execution of the system E. In other words, a system E is *BNWMC* if what a high level user sees of the system with respect to write actions is not modified by composing any low level process Π to E.

Definition 4.3 (BNWMC): Let $E \in \mathcal{E}$.

$$E \in BNWMC \text{ iff } \forall\ \Pi \in \mathcal{E}_L,\ E\,/_{H_r}\backslash_L \approx (E\,/_{H_r} \mid \Pi)\backslash_L$$

The following result is immediate:

Proposition 4.4: $BNMC \subset BNWMC$

Proof: To demonstrate $BNMC \subseteq BNWMC$ is enough to say that (weak) bisimilarity is preserved by relabelling [16]. Let $E \in \mathcal{E}$. Thus $E \in BNMC$ implies $\forall\ \Pi \in \mathcal{E}_L,\ E\backslash_L \approx (E|\Pi)\backslash_L$. Since bisimulation is closed under relabelling we obtain that $\forall\ \Pi \in \mathcal{E}_L,\ E\backslash_L\,/_{H_r} \approx (E \mid \Pi)\backslash_L\,/_{H_r}$. From $L \cap H_r = \emptyset$ we have that for all $E \in \mathcal{E}$ holds $E\backslash_L\,/_{H_r} = E\,/_{H_r}\backslash_L$ and we are done. The example of the introduction proves that the inclusion is strict; that is, $P \notin BNMC$ while $P \in BNWMC$, where $P \stackrel{\triangle}{=} wannaread_{lw}.read_{hr}.0 \mid write_{hw}.0$

■

Example 4.5: We want to establish, once we have weakened our notion of security, if intrusions are possible in this system; we obtain that the process $System$ does not satisfy *BNWMC*. To show this, we consider a low user which write both the file /etc/utmp and the mailbox.

$$U \stackrel{\triangle}{=} \overline{write}(utmp, L, pwd).U'$$
$$U' \stackrel{\triangle}{=} \overline{write}(mailbox, L, r0).0$$

Here $(System\,/_{H_r} \mid U)\backslash_L$ can do an internal move in which the user overwrites the file /etc/utmp with the content "/etc/passwd":

$$(\ System\,/_{H_r} \mid U\ )\backslash_L \quad \stackrel{\tau}{\rightarrow}$$
$$(\ (\ F_{pwd} \mid Comsat\ )\,/_{H_r} \mid U'\ )\backslash_L\ .$$

We indicate with the process F_{pwd} the agent reached from $Fs(s)$ after the updating of /etc/utmp.

After another internal move the user sends the mail with content "root::0:0:" ; we denote by F_{cmp} the compromised file system agent reached from the state F_{pwd} by updating the mailbox with the new message.

$$(\ (\ F_{pwd} \mid Comsat\ )\,/_{H_r} \mid U'\ )\backslash_L \quad \stackrel{\tau}{\rightarrow}\equiv$$
$$(\ F_{cmp} \mid Comsat\ )\,/_{H_r}\backslash_L$$

Now $(F_{cmp} \mid Comsat)\,/_{H_r}\backslash_L$ performs a τ transition in which comsat asks to read /etc/utmp; then, comsat receives the content of the mailbox through another internal transition. We remark that, differently from Example 4.2, the channel sending the content of the mailbox is now unobservable, since high read actions are hidden. Then, analogously, comsat through two internal transitions asks the content of the terminal file and receives the bogus content "/etc/passwd".

$$(F_{cmp} \mid Comsat)\,/_{H_r}\backslash_L \quad \stackrel{\tau}{\Longrightarrow}$$
$$(F_{cmp} \mid \overline{write}(pwd, H, r0).Comsat)\,/_{H_r}\backslash_L$$

Intuitively, the file system state is still represented by the process F_{cmp} since read requests do not determine an updating of the file system. Now we have that

$$(F_{cmp} \mid \overline{write}(pwd, H, r0).Comsat)\,/_{H_r}\backslash_L \stackrel{\overline{write}(pwd,H,r0)}{\longrightarrow}$$

while this action is unobservable after any number of τ reductions of $System\,/_{H_r}\backslash_L$, since $utmp$ was not containing the value pwd in the init state.

Applying our technique, we have thus found that the UNIX-like system model introduced is flawed, since a low user of the system may indirectly break the "no write up" mandatory system's security policy obtaining that the high password file is overwritten. Using a security property less discriminating than *BNMC* gave us the possibility to individuate the core of the intrusion. That is, by checking if the system satisfies the *BNWMC* property we individuate the action which exactly cause the security policy to be broken.

V. Final Remarks and Related Work

We have studied the Intrusion Detection problem on real systems governed by a mandatory security policy. Our thesis is that intrusion detection is a special case of access control, where an intrusion is the sequence of steps which breaks the security policy of the system. We have proposed a new security property named *BNWMC* which characterizes the security of CCS-like models of real systems w.r.t. the integrity of data.

We point out that *BNDC* verification techniques [26] can be used to prove both that a system is *BNMC* and that is *BNWMC*.

To the best of our knowledge, few works rely on a formal model expressed in a process algebra to analyze intrusions within a system, and none of these analyze information flows.

Ramakrishnan and Sekar's works [14], [27] without doubt inspired us. In [14], [27], the authors define security properties trough an intentions model which captures the intended outcome of executing every program: the system model has vulnerabilities if it contains path to unsafe states for which there exists no corresponding path in the intentions model. Their implementation find vulnerabilities as violations of invariant properties: for instance, they query the model checker to check the reachability of *_write([etc,passwd],_)*, a state where a process writes /etc/passwd. Path properties can be encoded in temporal logic. The analysis may be extended to infinite systems, but in this case a bounded-depth search need to be performed in order to ensure termination. In [27] the infinity condition is handled in the model checker by using term constraints and by abstracting the sequences to finite

(possibly repeating) segments of a certain kinds; moreover, the language is extended with object-oriented capabilities.

Rohrmair and Lowe [28] model a small network scenario using the CSP process algebra and consider the problem to discover attack strategies which blind network Intrusion Detection Systems. Their model consists in a signature based IDS process protecting a host, which is the target of the attack. The Intrusion Detection System is considered to be perfect and therefore knows all vulnerabilities that could be used to cause a security branch in the system. Quoting the authors: "In practice this is impossible because many vulnerabilities are not revealed yet. We have to make this assumption to generalize all current existing IDSs". Once the CSP model is defined, they check through a model checker if the set of all the traces of the model is a subset of the set of valid traces. The valid traces are defined as the traces of the specification process S, which should capture the intended behaviour of the model. For instance if $alert$ is the action raised by process IDS in the presence of an attack and $fail$ is a target action meaning that the target is compromised, then $S \stackrel{\triangle}{=} alert.fail.S$ indicate that the IDS should have a log-entry once a successful attack is performed. Now if $S \sqsubseteq_T Model$ does not hold, where $\sqsubseteq_T$ is the CSP refinement on traces, then (at least) a new attack which eludes the IDS is discovered.

ACKNOWLEDGMENTS

We are grateful to Sabina Rossi for precious comments and suggestions. Discussions on information flow with Riccardo Focardi, which revised the preliminary work [25], have helped us to improve the paper. Michele Bugliesi gave us fruitful advises about the interpretation of the intrusion detection problem. The author was partially supported by EU-FET project 'MyThS' IST-2001-32617.

REFERENCES

[1] J. P. Anderson, "Computer security threat monitoring and surveillance," Fort Washington, Pennsylvania, Tech. Rep. 79F296400, April 1980.

[2] D. E. Denning, "An Intrusion-Detection model," *IEEE Trans. on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.

[3] T. F. Lunt, "IDES: An intelligent system for detecting intruders," in *Proceedings of the Computer Security, Treat and Countermeasures Symposium (Rome, Italy)*, November 1990.

[4] T. Lunt and D. Anderson, "Software requirements specification: Next-generation Intrusion Detection expert system," SRI International, Computer Science Laboratory, Tech. Rep., 1994.

[5] K. Ilgun, R. A. Kemmerer, and P. A. Porras, "State transition analysis: A rule-based Intrusion Detection approach," *IEEE Transactions on Software Engineering*, vol. 21, pp. 1–22, March 1995.

[6] S. Kumar, "Classification and Detection of computer Intrusions," Ph.D. dissertation, Purdue University - COAST Laboratory ,West Lafayette, IN, USA, 1995.

[7] A. Mounji, "Languages and tools for rule-based distributed Intrusion Detection," Ph.D. dissertation, Facult es Universitaires, Notre-Dame de la Paix, Namur (Belgium), Institut d'informatique, September 1997.

[8] P. Ning, S. Jajodia, and X. S. Wang, "Abstraction-based intrusion detection in distributed environments," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, no. 4, pp. 407–452, 2001.

[9] S. T. Eckmann and G. Vigna, "STATL: An attack language for state-based Intrusion Detection," Reliable Software Group Department of CS - University of California - Santa Barbara, Tech. Rep., 2000.

[10] F. Cuppens and R. Ortalo, "Lambda: A language to model a database for detection of attacks," in *Proc. of Third International Workshop on Recent Advances in Intrusion Detection (RAID'2000)*, 2000.

[11] M. Roger and J. Goubault-Larrecq, "Log auditing through model checking," in *Proc. of 14th IEEE Computer Security Foundations Workshop (CSFW'01), Cape Breton, Nova Scotia, Canada, June 2001*. IEEE Comp. Soc. Press, 2001, pp. 220–236. [Online]. Available: citeseer.nj.nec.com/roger01log.html

[12] J.-P. Pouzol and M. Ducassè, "Formal specification of intrusion signatures and detection rules," in *Proc. of 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, 2002.

[13] M. Bishop, "Security problems with the unix operating system," Department of Computer Sciences, Purdue University, Tech. Rep., 1983.

[14] C. Ramakrishnan and R. Sekar, "Model-based vulnerability analysis of computer systems," in *Proc. of the 2nd Int. Workshop on Verification, Model Checking and Abstract Interpretation (VMCAI'98)*, 1998.

[15] C. Hoare, *Communicating Sequential Processes*, ser. International Series in Computer Science. Prentice Hall, 1985.

[16] R. Milner, *Communication and Concurrency*. Prentice-Hall, 1989.

[17] K. Biba, "Integrity considerations for secure computer systems," MITRE Corporation, Tech. Rep. ESD-TR-76-372, 1977.

[18] D. E. Bell and L. J. L. Padula, "Secure computer systems: Unified exposition and multics interpretation," MITRE MTR-2997, Technical Report ESD-TR-75-306, 1975.

[19] J. A. Goguen and J. Meseguer, "Security Policies and Security Models," in *Proc. of the IEEE Symposium on Security and Privacy (SSP'82)*. IEEE Computer Society Press, 1982, pp. 11–20.

[20] D. Sutherland, "A Model of Information," in *Proc. of the 9th National Computer Security Conference*, 1986, pp. 175–183.

[21] S. N. Foley, "A Universal Theory of Information Flow," in *Proc. of the IEEE Symposium on Security and Privacy (SSP'87)*. IEEE Computer Society Press, 1987, pp. 116–122.

[22] D. McCullough, "Specifications for Multi-Level Security and a Hook-Up Property," in *Proc. of the IEEE Symposium on Security and Privacy (SSP'87)*. IEEE Computer Society Press, 1987, pp. 161–166.

[23] J. McLean, "Security Models," *Encyclopedia of Software Engineering*, 1994.

[24] R. Focardi and R. Gorrieri, "Classification of Security Properties (Part I: Information Flow)," in *Proc. of Foundations of Security Analysis and Design (FOSAD'01)*, ser. LNCS, R. Focardi and R. Gorrieri, Eds., vol. 2171. Springer-Verlag, 2001, pp. 331–396.

[25] M. Giunti, "Modelli Formali per Intrusion Detection," Master's thesis, Dipartimento di Informatica, Università Ca' Foscari di Venezia, 2002.

[26] A. Bossi, R. Focardi, C. Piazza, and S. Rossi, "Verifying Persistent Security Properties," *Computer Languages, Systems and Structures*, vol. 30, no. 3-4, pp. 234–258, 2004.

[27] C. Ramakrishnan and R. Sekar, "Model-based analysis of configuration vulnerabilities," *Journal of Computer Security (JCS)*, vol. 1/2, no. 10, pp. 189–209, 2002.

[28] G. Rohrmair and G. Lowe, "Using CSP to detect insertion and evasion possibilities within the intrusion detection area," in *Proc. of BCS Workshop on Formal Aspects of Security (BCS '02).*, 2002.

Using Game Theory in Stochastic Models for Quantifying Security

Karin Sallhammar and Svein J. Knapskog
Centre for Quantifiable Quality of Service in Communication Systems
Norwegian University of Science and Technology,
Trondheim, Norway
{sallhamm, knapskog}@q2s.ntnu.no

***Abstract*— In this paper, game theory is suggested as a method for modelling and computing the probabilities of expected behaviour of attackers in a quantitative stochastic model of security. The stochastic model presented her is very simple, modelling a penetration attempt as a series of intentional state changes that lead an ICT system from an assumed secure state to a state where one or more of the systems security aspects are compromised. The game situation models the actions of the attacker under the condition that at each intermediate stage of the attack, the attempt may be detected and measures taken by the system owner to bring the system back to the originating secure state. Assumptions are made for the possible rewards for the players of the game, allowing the calculation of the mean time to first security breach (MTFSB) for the system. An example of the possible use of the model is provided by calculating the MTFSB for a root privilege attack on a UNIX system.**

***Index Terms*— Computer security, quantification, stochastic analysis, attack models, game theory**

I. Introduction

The security of operating computer systems has traditionally only been expressed in a qualitative manner. However, to be able to offer a security dimension to QoS architectures, it is important to find quantitative measures of security. In the dependability community, methods for quantifying reliability, availability and safety, are well-known and effective. By using state space modelling methods, operational measures for the system, such as mean time between failures (MTBF), mean time to failure (MTTF) or mean time to first failure (MTFF), can be computed. During the past decade, some research on applying the dependability paradigm to security has been performed, using an analogy between system failure and security breach, aiming and attempting to quantify security by calculating measures such as mean time to security compromise.

However, in contrast to failures, attacks may not always be well characterized by models of a random nature. Most attackers will act with an intent and consider the possible consequences; satisfaction, profit and status versus effort and risk of the actions before they act. This paper uses a game theoretic approach to model the expected behavior of attackers for use in stochastic modelling techniques.

The gain of using a game theoretic approach in a security related context is twofold. Firstly, we believe it can provide a more accurate model of the attackers' expected behavior, which can be used to assign more realistic transitions probabilities in the stochastic models. Secondly, it may also help administrators to find the optimal defense strategies of a system and to calculate the expected loss associated with different defense strategies. This work is a demonstration of the former application.

In order to keep the focus on the game theoretic model, issues relating to model parametrization is ignored. Therefore only guessed values are used to demonstrate how the model can be used to obtain quantitative measures.

II. Related Work

There are several papers on quantification of security. In [7], a first step towards operational measures of computer security is discussed. The authors point to the lack of quantitative measures for determining operational security and relate security assessment to the dependability domain. Quantitative measures such as mean effort to security breach (MESB) are defined and discussed. [12] presents a quantitative model to measure known Unix security vulnerabilities using a privilege graph, which is transformed into a Markov chain. The model allows for the characterization of operational security expressed as mean effort to security failure as proposed by [7]. Further, in [10], [16] and [2] traditional stochastic modelling techniques are used to capture attacker behavior and the system's response to attacks and intrusions. A quantitative security analysis is carried out for the steady state behavior of the system.

Game theory in a security related context has also been utilized in previous papers. In [1], a model for attacker and intrusion detection system (IDS) behavior within a two-person, nonzero-sum, noncooperative game framework is suggested. The possible use of game theory for development of decision and control algorithms is investigated. In [9], a game theoretic method for analysing the security of computer networks is presented. The interactions between an attacker and the administrator are modelled as a two-player stochastic game for which best-response strategies (Nash Equilibrium) are computed. In [8] a preliminary framework for modelling attacker intent, objectives and strategies (AIOS) is presented. To infer AIOS a game theoretic approach is used and models for different threat environments are suggested.

Based on the game theoretic work of [8], [1] and [9], a method to model and compute the probabilities of malicious user actions for use in stochastic models is suggested. To demonstrate how to use the method, a real-world example of an attack against a system is modelled, the optimal strategy of the attack is calculated and, following the approach of [10], [2] and [12], the quantitative measure mean time to first security breach (MTFSB) is obtained for the system.

III. The Stochastic Model

Analogously to dependability analysis where system failure is a concept denoting the system's inability to deliver its services, in the security community one often talks of *security breach*; a state where the system deviates from its security requirements. A security breach might accidentally be caused by normal usage operation, but more likely by *intentional attacks* upon the system. Such attacks on an operating computer system can often be modelled as a series of state changes of the system that lead from an initial secure state to one or more target compromised states, i.e. security breach states. A successful attack against the system may therefore consist of many subsequent elementary attack actions. At each intermediate stage of the attack, the attacker will therefore have the choice of either

Attack by performing the next elementary step in the attack.
- If the attacker succeeds the system will be transferred from state i to state $i+1$.
- If the attacker fails the system will remain (temporary) in state i.

Resign and interrupt the ongoing attack
- The system will be remain (temporary) in state i.

On the other hand, at each intermediate stage, the system administrator may

Detect the attack and bring the system back to a secure state
- The system will be transferred from state i to state 0, hence, the attacker will not have the possibility of continuing the attack.

This is illustrated in Fig. 1. In the model it is assumed that once an attack is initiated, the attacker will never voluntarily try to revert the system to any of the previous states. The model also assumes there is only one single path to the security breach state; a somewhat simplified view of reality.

A. Sojourn Time

Since the state transition model presented in Fig. 1 is stochastic by nature, the calender time spent in each state of the system model will be a random variable. The time or effort taken for an attacker to cause a transition will depend on several factors; the attacker's knowledge and background, robustness of the system etc. See e.g. [3] for a thorough discussion on this topic and [6] for empirical data collected from intrusion experiments. As mentioned in the introduction, to keep the focus on how to apply game theory in stochastic models, the differences between time and effort and the problems regarding finding suitable distributions and parameterizing the model will be ignored in this paper.

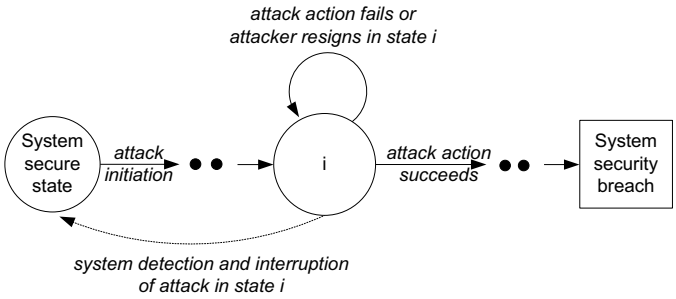


Fig. 1. Penetration of a computer system modelled as a series of state changes

For simplification, we therefore model the time between two subsequent attacks are initiated against the system as an negatively exponentially distributed (n.e.d.) variable with parameter λ_{attack}, i.e.

$$P(t) = 1 - exp(-\lambda_{attack} t) \tag{1}$$

Once an attack is initiated, the initial secure system will be transferred into the subsequent state $i = 1$. As an attack has been initiated, the time needed for the attacker to perform the next elementary step of the attack when the system is in state i, and the corresponding time needed for the system to detect and interrupt the ongoing attack during state i, are also modelled by the n.e.d, i.e.

$$P_{attack(i)}(t) = 1 - exp(-\lambda_i t) \tag{2}$$

and

$$P_{detect(i)}(t) = 1 - exp(- \ _i t) \tag{3}$$

respectively. Thus, $\frac{1}{\lambda_i}$ and $\frac{1}{\ _i}$ will be the respective mean time an attacker and the system spend in state i of the model before causing a transition. The two "competing" processes with rates λ_i and $\ _i$ representing the attacker and system actions in state i can be merged into one Poisson process, hence, due to the memoryless property of the exponential distribution, the state transition model then will be transformed into a continuous time Markov chain (CTMC) [14] with discrete state space, formally described as:

$$\{X(t) : t \geq 0\}, X_s = \{0, 1, 2, .., n\}. \tag{4}$$

for which analytic analysis is possible. This model during the given assumptions is displayed in Fig. 2.

In reality, there may be other types of distributions than the negative exponential one, which are more suitable to model the transitions of the stochastic model. However, to facilitate analytic analysis the n.e.d. was chosen for all transitions in the stochastic model.

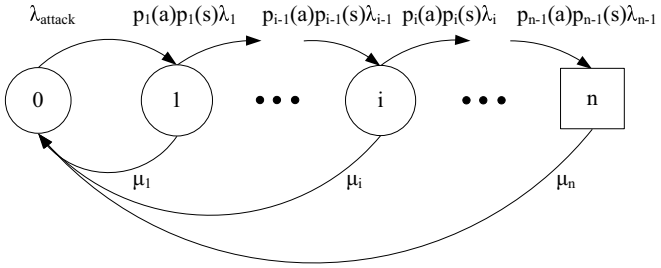


Fig. 2. A general stochastic model for penetration of a system.

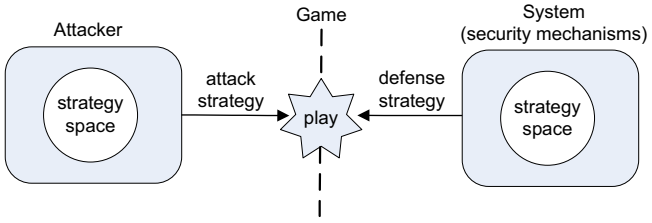


Fig. 3. The interactions between an attacker and the system modelled as a 2-player static game

B. Transition Probabilities

As previously mentioned, in each intermediate state i an attacker has two possible choices of action; with probability $p_i(a)$ he will decide to continue the attack and with probability $1-p_i(a)$ he will resign and interrupt the attack. This *decision probability* represents an important difference between dependability and security analysis when using stochastic modelling techniques. In traditional dependability analysis only accidental failures are modelled; there is no human decision involved in the occurrence of a failure, hence $p_i(a) = 1$, for all i. However, when modelling attacks rather than failures one must keep in mind that an attacker will consider the consequences of his actions and compare the possible payoff versus the risk of each elementary attack action. An attacker may therefore choose to interrupt an ongoing attack at a certain stage or to not start the attack at all. Therefore, for each transition representing an elementary step in the attack $p_i(a)$ should be explicitly calculated.

To be able to bring the system closer to the security breach state, an attacker not only has to decide upon an action, but he must also *succeed* with the particular action. This probability of success is denoted by $p_i(s)$ and is also included in the stochastic model presented in Fig. 2.

Using the attacker's and system's action rates together with the transition probabilities, the *instantaneous* transition rates between the state i and $i+1$ in the stochastic model can be calculated as

$$q_{i,i+1} = p_i(a)p_i(s) \cdot \lambda_i, \tag{5}$$

and between state i and 0 as

$$v_{i,0} = \quad _i \tag{6}$$

As will be demonstrated in Section V, the instantaneous transition rates can be used for quantitative security analysis of the operating system.

IV. The Game Model

To determine the decision probabilities $p_i(a)$ game theory can be used. (A formal definition is given in Appendix). If one views each elementary attack action causing a transition in the stochastic model as an action in a game, where the attacker's choices of action is based on intelligent considerations of the possible consequences, then the interactions between the attacker and the system can be modelled as a 2-player static game. This is displayed in Fig. 3

The complete action set for the game then includes the attacker's two choices of action together with the possible consequences, i.e. for each state i:

a_i is the elementary attack action bringing the system from state i to $i+1$,
r_i is the resignation of the attack in state i,
d_i represents that the elementary attack action a_i will be detected by the system.
ϕ_i represents that the elementary attack a_i action will be undetected.

Hence, for each state i there is a game model $G(i)$ defined by

$$\begin{aligned} &N = \{1,2\} = \{\textit{attacker, system}\},\\ &A_i = \{a_i, r_i, d_i, \phi_i\},\\ &u_i = \begin{array}{|c|c|c|}\hline & d_i & \phi_i \\ \hline a_i & u_{i1} & u_{i2} \\ \hline r_i & u_{i3} & u_{i4} \\ \hline \end{array} \end{aligned} \tag{7}$$

where $u_i = \{u_{i1}, u_{i2}, u_{i3}, u_{i4}\}$ is the payoff received by the attacker for each possible combination of action and response from the system

For the game defined in (7) it is possible to use (18) in Appendix to calculate the attacker's expected payoff for a choice of action

$$\begin{aligned} u_i(a_i) &= p(d_i) \cdot u_{i1} + p(\phi_i) \cdot u_{i2},\\ u_i(r_i) &= p(d_i) \cdot u_{i3} + p(\phi_i) \cdot u_{i4}. \end{aligned} \tag{8}$$

Since in most cases an attacker *do not know* the exact probability that his action will remain undetected, game theory says he should assume that his opponent (the system) is a conscious player of the game which seeks to minimize the attacker's expected payoff [15], hence, the minimax solution of the particular game $G(i)$ can be calculated as

$$\alpha_i^* = \min_{\alpha_i(d_i)\alpha_i(a_i)} \max \{u_i\big(\alpha_i(a_i)\big), u_i\big(\alpha_i(r_i)\big)\} \tag{9}$$

as defined in Appendix. In reality, since the reward experienced by the attacker from an outcome rarely coincide with the system's loss, the game is usually not truly zero-sum. However, defining payoff values for the system is irrelevant in game models like these where the attacker is the only player who is capable of making intelligent decisions.

The minimax solutions $\alpha_i^*(a_i)$ of the games $G(i)$ for all the elementary attack actions $i = 1, .., (n-1)$ represents a complete attack strategy which has the property that, by following it, an attacker will know that he has maximized his expected payoff of the attack. This gives him a guarantee of the result from the attack regardless of if one of his elementary attack actions will be detected by the system or not; the "no regrets property" of game theory. Several experiments indicate that this search for guarantees is a very strong motivator of human behavior and assuming that the attacker population targeting the system will make rational choices relative to their objectives, the situation will in the long run naturally gravitate towards the minimax solution (i.e. the Nash equilibrium) [4], [15]. The minimax strategy will therefore indicate how *rational* attackers will behave.

The attacker decision probability in state i of the stochastic model can therefore be directly derived from the minimax solution of the corresponding game as

$$p_i(a) = \alpha_i^*(a_i) \tag{10}$$

Note that, when α_i^* is the solution of a *static* game, it is only correct to use (10) for the stochastic model as long as the decision probability $p_i(a)$ depends on the current state i in the stochastic model only (i.e. the Markov property holds).

V. Quantitative Analysis

Returning to the stochastic model presented in Section III and illustrated in Fig. 2, since the stochastic model is a homogenous continuous time Markov chain [14], it is straightforward to determine the limiting probabilities of each state in the model. By solving the equation system

$$\begin{cases} P_{S0} \cdot \lambda_{attack} = P_{S1} \cdot (q_{12} + v_{10}) \\ P_{S1} \cdot q_{12} = P_{S2} \cdot (q_{23} + v_{20}) \\ \vdots \\ P_{Sn\ \ 1} \cdot q_{n\ \ 1,n} = P_{Sn} \cdot v_{n0} \\ \sum_{i=0..n} P_{Si} = 1 \end{cases} \tag{11}$$

one can obtain an expression for P_{Sn}; the stationary probability of being in the security breach state. Now, one can use traditional dependability techniques (see e.g. [13] or [5]) to compute quantitative measures of the system's operational security. Using the approach outlined in [5], the mean time to first security breach (MTFSB) for our model can be computed as:

$$MTFSB = \frac{1 - P_{Sn}}{P_{Sn} \cdot v_{n0}} \tag{12}$$

Hence, by parameterizing the model, solving (11) and using (12), MTFSB can easily be obtained for the system. The MTFSB measure provides the mean time it takes before the system reaches its defined security breach state for the first time. For a system, which starts in a initially secure state, MTFSB will be a quantitative measure of the system's operational security when considering a certain kind of attack posed upon the system.

As previously mentioned, using other types of distributions than the n.e.d. will exclude many of the well-known methods for analytic analysis of the model, however, simulation can then be used to obtain the MTFSB for the system.

VI. Application

A. Example of an Attacker-System Game Model

A typical example of how an attacker may experience the outcome of his two choices in state i is

$$u_i = \begin{array}{|c|c|c|} \hline & d_i & \phi_i \\ \hline a_i & u_{i1} & u_{i2} \\ \hline r_i & u_{i3} & u_{i4} \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline & \text{detected} & \text{undetected} \\ \hline \text{attack} & -1 & 2 \\ \hline \text{give up} & 0 & -1 \\ \hline \end{array} \tag{13}$$

If the attacker chooses to perform the elementary attack action, without being detected, he receives a positive payoff ($u_{i2} = 2$). But if he performs the action and the system will detect this kind of violation, he receives a negative payoff ($u_{i1} = -1$). On the other hand, if the attacker chooses to resign the attack, even though he would not been detected if he had tried, he also receives a negative payoff ($u_{i4} = -1$). However, if he chooses to resign when he would have been detected if he tried, no payoff is received ($u_{i3} = 0$).

The attacker's expected payoff $u_i(a_i)$ as a function of the probability $\alpha_i(a_i)$ of trying the attack action for this game is illustrated in Fig. 4. The dashed line displays the expected payoff when the system always detects the attack action (i.e. $\alpha_i(d_i) = 1$) whereas the dotted line displays the expected payoff if the system never detects the attack action (i.e. $\alpha_i(d_i) = 0$). Hence, the strategy which provides the attacker with the highest expected payoff is the minimax solution of this game

$$p_i(a) = \alpha_i^*(a_i) = 0.25 \tag{14}$$

as indicated the Fig. 4 and verified by the Gambit software tool [11].

B. Example of Quantification of an Attack

In a root privileges attack, an attacker tries to obtain root access to a Unix or Linux system connected to a network.

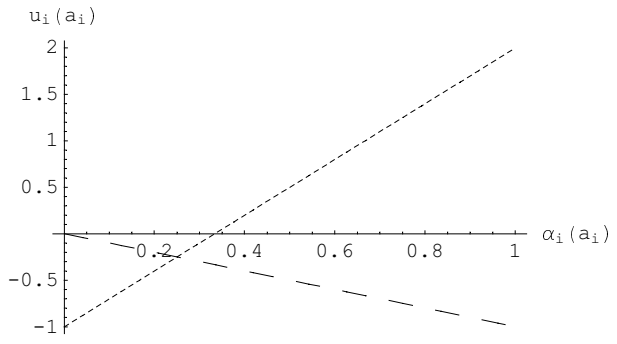


Fig. 4. The expected payoff for the game in Table 13.

TABLE I
NUMERICAL VALUES FOR THE PARAMETERS OF THE MODEL IN FIG. 5

Parameter	Value (s)
$1/\lambda_{attack}$	$9.0 \cdot 10^5$
$1/\lambda_1$	$1.8 \cdot 10^4$
$1/\lambda_2$	$2.2 \cdot 10^4$
$p_1(s)$	0.9
$p_2(s)$	0.7
$1/\mu_2$	$8.5 \cdot 10^4$
$1/\mu_3$	$3.6 \cdot 10^3$

Assuming that the attacker is not an insider (a registered user), one common way to gain root access is by

1) crack or sniff passwords to get access to local user account
2) trigger a local exploit, e.g. the mremap() exploit on Linux, to get root privileges

The stochastic model for this attack scenario is displayed in Fig. 5. For security quantification the model has been parameterized with the values indicated in Table I. To shorten the example, the attacker's payoffs for attack step 1 and 2 are both assigned values as in the example (13) with the minimax solution $\alpha_i^*(a_i) = 0.25$ (14), which gives $p_1(a) = p_2(a) = \alpha_i^*(a_i) = 0.25$.

By inserting the values from Table I and solving the equation system

$$\begin{cases} P_{S0} \cdot \lambda_{attack} & = P_{S1} \cdot p_1(a)p_1(s)\lambda_1 \\ P_{S1} \cdot p_1(a)p_1(s)\lambda_1 & = P_{S2} \cdot \left(p_2(a)p_2(s)\lambda_2 + \quad _2\right) \\ P_{S2} \cdot p_2(a)p_2(s)\lambda_2 & = P_{S3} \cdot \quad _3 \\ P_{S0} + P_{S1} + P_{S2} + P_{S3} & = 1 \end{cases} \tag{15}$$

we obtain an expression for P_{S3} , hence, by using (12), MTFSB for this type of attack is calculated as

$$MTFSB = \frac{1 - P_{S3}}{P_{S3} \cdot \quad _3} = ... = 2.555 \cdot 10^6 \text{ (sec)} \approx 30 \text{ (days)}. \tag{16}$$

The MTFSB measure of 30 days reflects how long one can expect that the system will remain secure from illegal root access when attacked by non-registered users during the given assumptions.

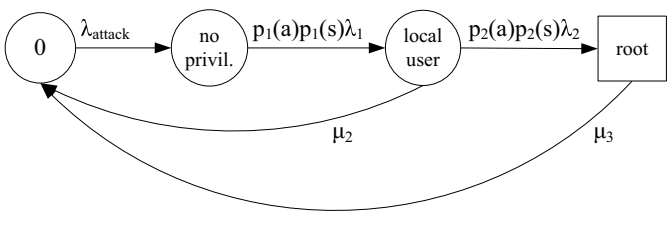


Fig. 5. A stochastic model for obtaining root access of a Linux/Unix system.

VII. CONCLUSIONS AND FURTHER WORK

In this paper, game theory is suggested as a method for modelling and computing probabilities of the expected behavior of attackers in quantitative stochastic models of security. One example of an attack against a system is modelled and the operational measure "mean time to first security breach" (MTFSB) is computed for the attack.

The model presented here is very simple. Further work will therefore include extending the game theoretic model with different attacker profiles; not all attackers will experience equal payoffs and some attackers tend to take more risks than others. Models including more than one type of attack and where there are more than one possible way to reach the security breach state, resulting in more complex attack graphs, will be developed. To avoid state space explosion in larger models, the possibilities of using stochastic Petri nets as a modelling and analyzing tool will be considered.

Regarding the game theoretic model, it is interesting to note that if the attacker knows the probability of getting caught at a certain stage of the attack, then there will always be a pure strategy (either to always attack or to always resign) that maximizes his expected received payoff. Also, in cases where the attacker does not know the exact probability of getting caught there might be other strategies than the minimax solution which gives him a larger payoff. However, as discussed in [15] when leaving the minimax strategy the attacker looses his guarantee of expected payoff and taking such risks seems contrary to human nature!

Furthermore, it can be argued that the players in this game (the attackers) may be unaware of, or ignore, the fact that they are playing a repeated game, hence, statistics of attacker behavior may not always converge to equilibrium in practice. A "one-shot game" with a pure minimax solution may therefore in many cases be more appropriate for modelling expected behavior of attackers. Whether the game model presented in this paper gives a realistic model of real world security related attacks will require further research including validation of the model against empirical data.

REFERENCES

[1] T. Alpcan and T. Basar. A game theoretic approach to decision and analysis in network intrusion detection. In *Proc. of 42nd IEEE Conference on Decision and Control*, 2003.
[2] K. B. B. Madan, K. Vaidyanathan Goseva-Popstojanova, and K. S. Trivedi. A method for modeling and quantifying the security attributes of intrusion tolerant systems. In *Performance Evaluation*, volume 56, 2004.
[3] S. Brocklehurst, B. Littlewood, J. Olovsson, and E. Jonsson. On measurement of operational security. *Aerospace and Electronic Systems Magazine, IEEE*, 9, 1994.
[4] R. Gibbons. *Game Theory for Applied Economists*. Princeton University Press, 1992.
[5] B. E. Helvik. *Dependable Computing Systems and Communication, Design and Evaluation*. 2003. Draft lecture notes for the course TTM4120 Dependable Systems, NTNU Norway.
[6] E. Jonsson and T. Olovsson. A quantitative model of the security intrusion process based on attacker behavior. *IEEE Trans. Software Eng.*, 4(25):235, April 1997.

[7] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, McDermid J., and D. Gollmann. Towards operational measures of computer security. *Journal of Computer Security*, 2:211–229, Oct 1993.

[8] Peng Liu and Wanyu Zang. Incentive-based modeling and inference of attacker intent, objectives, and strategies. In *Proceedings of the 10th ACM conference on Computer and communication security*, 2003.

[9] K. Lye and J. M. Wing. Game strategies in network security. In *Proc. of 15th IEEE Computer Security Foundations Workshop*, 2002.

[10] B.B. Madan, K. Vaidyanathan, and K.S. Trivedi. Modeling and quantification of security attributes of software systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)*, 2002.

[11] McLennan Andrew M. McKelvey, Richard D. and Theodore L. Turocy. Gambit: Software tools for game theory, version 0.97.0.6, 2004. http://econweb.tamu.edu/gambit/.

[12] R. Ortalo and Y. Deswarte. Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Transactions on Software Engineering*, 25(5):633–650, Sept/Oct 1999.

[13] A. Puliafito R. A. Sahner, K. S. Trivedi. *Performance and Reliability Analysis of Computer Systems*. Kluwer Academic Publishers, 1996.

[14] S. M. Ross. *Introduction to Probability Models*. Academic Press, 8 edition, 2003.

[15] S. Stahl. *A Gentle Introduction to Game Theory*. American Mathematical Society, 1991.

[16] D. Wang, B.B. Madan, and K.S. Trivedi. Security analysis of sitar intrusion tolerance system. *ACM SSRS'03*, 2003.

APPENDIX
GAME THEORY

Formally, a static n-player game in strategic form with complete information is a 3-tuple $(N, \{A_k\}_{k\in N}, \{u_k\}_{k\in N})$. The game consists of

- a set of players: $N = 1, ..., n$,
- the action (strategy) spaces of players: A_k,$k = 1, .., n$ where A_k is the set of all available actions to player k. The outcome space is then defined as $A = \times_{k\in N} A_k = \{(a_1, .., a_n) : a_k \in A_k, k = 1, .., n\}$ and is thus nothing but an action profile.
- the payoff function of players: $u_k : A \rightarrow \mathbf{R}, k = 1, .., n\}$

If N and A_k are finite, the game is called *finite*. In particular, if the game is played by only two players ($N = \{1, 2\}$) exactly four pieces of information is needed to uniquely define the game: (A_1, A_2, u_1, u_2).

A (pure) *strategy* for a player is a choice of action from his set of available actions, whereas

Definition 1 A **mixed strategy** α_k for player k, is a probability distribution over his set of available actions, A_k, i.e. if player k has m actions available, a mixed strategy is an m dimensional vector

$$(\alpha_k^1, \alpha_k^2, .., \alpha_k^m) \text{ such that } \alpha_k^\beta \geq 0 \text{ for all } \beta = 1, 2, .., m, \text{ and } \sum_{\beta=1}^{m} \alpha_k^\beta = 1 \tag{17}$$

Hence, $\alpha_k(a_k^\beta)$ is the probability that player k will take action a_k^β.

The *payoff* function maps the action profile of a player to the corresponding consequence (reward or loss) experienced by the player. If each outcome $a \in A$ occurs with probability $p(a)$, then the *expected payoff* of player k is

$$u_k(p) = \sum_{a\in A} p(a)u_k(a) \tag{18}$$

In a nonzero-sum game, a *rational* player will always try to maximize her own expected payoff from the game. The choice of a_k that maximize player k's expected payoff over her action space A_k is called the player's *best response* action. The decision making of player k in a game then becomes

$$\underset{a_k\in A_k}{max}\ u_k(a_k, a_{-k}) \tag{19}$$

where a_{-k} is the action choice of the other players, unknown by player k.

Definition 2 The **best response correspondence** of player k is the set of mixed strategies which are optimal given the other player's mixed strategies. In other words:

$$B_k(\alpha_{-k}) = \underset{\alpha_k\in\Delta(A_k)}{\arg\max}\ u_k(\alpha_k, \alpha_{-k}) \tag{20}$$

Definition 3 A **mixed strategy equilibrium** (Nash equilibrium) of a game G in strategic form is a mixed strategy profile $(\alpha_1^*, .., \alpha_n^*)$ such that, for all $k = 1, .., n$

$$\alpha_k^* \in \underset{\alpha_k\in\Delta(A_k)}{\arg\max}\ u_k(\alpha_k, \alpha_{-k}^*) \tag{21}$$

or

$$\alpha_k^* \in B_k(\alpha_{-k}^*) \tag{22}$$

In a zero-sum two-player game where one player's gain is the other player's loss the Nash equilibrium of the game is also called the *minimax* solution of the game.

A technique for modeling and analysis of Common Criteria security environments and security objectives

Jussipekka Leiwo

***Abstract*— Common Criteria provides a standardized and internationally recognized framework for developing high assurance IT security products. As part of the Security Target, developer specifies security objectives for the product and for the intended operational environment of the product. Being textual artifacts, specifications are often intuitive and hard to assess for semantical correctness. While fulfilling the Common Criteria requirements for content and presentation, they fail to appropriately capture the security problem addressed. A modeling technique shall be proposed for defining security environments. Applicability of the proposed technique shall be demonstrate by using it to assess the well known Secure Signature–Creation Device Protection Profile.**

***Index Terms*— Common Criteria, Security specifications, Security specification analysis, Secure Signature-Creation Device**

I. Introduction

European Union Directive 1999/93/EC [8] states the requirements for member states to adapt national legislations so that digital signatures on electronic documents are recognized as equivalent to hand written signatures on paper documents. A Common Criteria [1], [2], [3] Protection Profile for Secure Signature-Creation Devices [14] (SSCD PP) has been approved as a technical annex to the directive [7] so any device evaluated to meet the SSCD PP that implements a set of approved signature computation algorithms (enumerated in [9]) must be approved as a valid SSCD in any member state. Member states may set additional criteria for nationally approved SSCDs but these devices are not necessarily approved in other member states. Signatures computed with a device approved in any member state, however, must be recognized in all member states.

An evaluated product conforming to a Protection Profile (PP) must preserve each security objective, implement security functionality to address each security functional requirement (SFR), and provide assurance evidence to demonstrate that the product meets the evaluation assurance level (EAL) defined in the Protection Profile. In addition to meeting the requirements for content and presentation stated in assurance class *APE: Protection Profile Evaluation*, a protection profile should be unambiguous to facilitate specification of Security Targets (ST) that capture both the letter and the intent of the protection profile. SSCD PP is evaluated and certified to meet Common Criteria requirements and a number of conformant smart cards have been engineered by major manufacturers. A rigorous analysis, however, exposes inadequacies in the SSCD PP and demonstrates that many interpretations the developer must make when designing a ST are not obvious from the PP. Therefore, it is possible for a developer to engineer a smart card that is evaluated to meet the SSCD PP but that fails to preserve the intended security objectives of a Secure Signature-Creation Device.

An analysis demonstrates inadequacies in the SSCD PP and in the Common Criteria treatment of assets and security objectives. Security objectives are fundamental to a security evaluation of a product as they define the scope of evaluation, hence must be kept to the minimum to facilitate cost efficiency, but also allow developer to change the focus of risk analysis to security requirements and measures to prevent foreseeable threats. While Security Functional Requirements appear as most complex PP or ST specification, an experienced developer can define them in a reasonably straightforward manner given well stated security objectives. Therefore, the focus of this analysis is on the security objectives. Alternative specifications are suggested for the SSCD PP and extensions to Common Criteria to increase accuracy of evaluations. Many dilemmas originate from the peculiarities of Common Criteria and are not specific to the SSCD PP. In addition to strengthening Common Criteria, findings also assist in any Protection Profile or Security Target specification and provide insights to the development of a SSCD.

Nanyang Technological University, School of Computer Engineering, Nanyang Avenue Block N4, Singapore 639798, asjleiwo@ntu.edu.sg

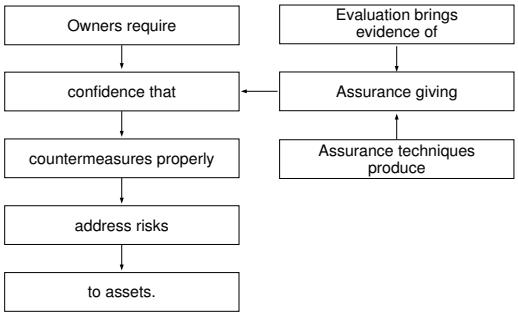


Fig. 1. Security countermeasures and assurance

II. Background

A. Security and assurance

Practical attacks against products and systems are usually against design and implementation of security techniques, not against number theoretical or other abstract properties of the techniques. Consequently, claims about the security of products and systems can not only be made with respect to security techniques implemented. Security technique specifications must be supported by claims on assurance that can be assessed to estimate the trustworthiness of a product or system. Claims are statements of the rigor to which specified assurance techniques are applied in the development of security features of a product or system (Fig. 1).

Consider a device for computing digital signatures required to preserve confidentiality and integrity of private keys stored on the device and to compute unforgeable signatures. Not only must cryptographic algorithms and parameters (key sizes, padding methods, etc.) be recognized but convincing assurance evidence must be produced by the developer to demonstrate that the device is engineered in a manner that key storage and signature computation are free from obvious vulnerabilities. Assurance evidence demonstrates, for example, that security requirements are properly formulated and capture the essence of security for the product, that analysis and design documents correctly and comprehensively capture those requirements and are revised into an appropriate implementation, that thorough functional and security testing is performed, that development environment is such that attempts to implant malicious code to the product are likely to be detected, that delivery procedures convince clients of receiving an authentic product that can be booted in a secure state,

that guidance for administrators and end users is comprehensive and that the product is assured to remain in a secure state if the guidance is followed.

Any disciplined engineering methodology may produce appropriate assurance evidence [13] but standards for assurance exist. Assurance may concern correctness or suitability of a product and may focus on the deliverables, processes or operational environment. From assurance techniques focusing on the correctness and suitability of deliverables, i.e. final product and associated documentation, various evaluation criteria are most widely applied.

Common Criteria (CC, [1], [2], [3]) is completed by a Common Evaluation Methodology (CEM, [5], [6]) and recognized through international arrangements [4], [12]. While more countries ratify recognition arrangements for Common Criteria. more Protection Profiles are specified, and more products are evaluated, it is safe to consider Common Criteria as a globally recognized assurance standard suitable for a framework in academic studies. Some familiarity with Common Criteria is expected from reader and an introduction to basic concepts is omitted due to space constraints.

Requirements for a PP are stated in assurance class *APE: Protection Profile Evaluation* and for a ST in assurance class *ASE: Security Target Evaluation*. They dictate in detail the content and presentation of a PP or ST and are quite identical. Developer performs a risk analysis and document the results in the statement of the TOE security environment. Security Environment is a specification of the context and environment of the intended use of the TOE and a statement of assumptions under which the TOE is secure, threats that must be countered by the security functionalities of the TOE, and organizational security policies that must be enforced in the operational environment to facilitate secure operation of the TOE. Security objectives are defined to counter the foreseeable threats. Security functional requirements are identified to constitute the security functionality required to preserve the security objectives and the EAL chosen. A TOE Summary Specification is established to characterize the security functions and assurance measures of the TOE and an optional justification of Protection Profile claims given.

B. Security environment and security objectives

Security environment specification must meet the requirements of assurance family *ASE_ENV.1* and security objectives specification must meet the requirements of assurance family *ASE_OBJ.1*. Security environment specification must state assets protected by the TOE and subjects interacting with the TOE. It must then include a statement of assumptions, threats and organizational security policies (OSP) relevant to the TOE.

Security objectives are defined for the TOE and for the environment of the TOE. The environment consists of the IT-environment and non-IT environment of the TOE. IT environment refers to any IT devices or systems relevant to secure operation of TOE, for example, a certificate generation application (CGA) that produces qualified certificates. Non-IT environment refers to processes and tasks carried out by human users of the TOE that affect the ability of the TOE to preserve it's security objectives. For example, an objective may state that a suitable training and guidance is given to the parties operating the TOE or IT-environment thereof. Developer also provides a security objectives rationale to demonstrate that the defined security objectives are relevant to the TOE and suitable to cover identified threats.

The content and presentation requirements for security environment and security objectives are identical for Protection Profiles and Security Targets. However, the idea is that a ST extends a PP by taking the elements defined in the PP as a baseline and adding components as relevant to the specific TOE and implementation technique. For conformance to a PP, security environment and security objectives are often incorporated to a ST without modifications. If no suitable PP exists, then ST developer has to define the security environment and security objectives from scratch for the particular TOE only.

C. Secure Signature-Creation Device Protection Profile

The particular concern is a SSCD Type 3, a device that can generate cryptographic key pairs and compute undeniable digital signatures. For technology independence, the directive and SSCD PP discuss electronic signatures and advanced electronic signatures but, in practice, digital signatures are the only technique to implement advanced electronic signatures. A SSCD Type 3 conformant to the SSCD PP concerns with the protection of the following assets.

- Signature Creation Data (SCD) refers to the private key used to perform an electronic signature generation. Confidentiality of the SCD must be maintained.
- Signature Verification Data (SVD) is the public key linked to the SCD and used to perform an electronic signature verification. Integrity of the SVD when exported to the Certificate Generation Application (CGA) must be maintained.
- Data To Be Signed (DTBS) and representation therefore represents the set of data, or its representation internal to the TOE, which is intended to be signed. Integrity of the data and representation must be maintained.
- Verification Authentication Data (VAD) is the PIN code or biometrics data entered by the end user to perform a signature operation. Confidentiality and authenticity of the VAD must be preserved as needed by the authentication method employed.
- Reference Authentication Data (RAD) is the reference PIN code or biometrics authentication reference used to identify and authenticate the end user. Integrity and confidentiality must be maintained.
- Signature-creation function of the SSCD using the SCD, quality of which must be maintained so that it can participate to the legal validity of electronic signatures.
- Unforgeability of electronic signatures computed with a SSCD using SCD must be assured.

The practice of naming assets is not as precise as the practice of naming other ST artifacts. To clarify the analysis by indicating that certain artifacts refer to these particular assets, above assets are named AST.SCD, AST.SVD, AST.DTBS, AST.VAD, AST.RAD, AST.SIG and AST.ES, in a corresponding order, in the subsequent analysis. More precise definitions are suggested in Sec. IV.

The following security objectives, to which alternative definitions are suggested in Sect. IV, for the TOE must be preserved:

OT.EMSEC_Design	Physical emanations security
OT.Lifecycle_Security	Lifecycle security of the TOE
OT.SCD_Secrecy	Secrecy of the signature-creation data
OT.SCD_SVD_Corresp	Correspondence between SVD and SCD
OT.SVD_Auth_TOE	TOE ensures authenticity of the SVD
OT.Tamper_ID	Tamper detection
OT.Tamper_Resistance	Tamper resistance
OT.Init	SCD/SVD generation
OT.SCD_Unique	Uniqueness of the signature-creation data
OT.DTBS_Integrity_TOE	Verification of the DTBS-representation integrity
OT.Sigy_SigF	Signature generation function for the legitimate signatory
OT.Sig_Secure	Cryptographic security of the electronic signature

Since assumptions and OSPs are not in the core of the analysis, they are omitted. Only exception is P.Sigy_SSCD that is relevant to the analysis as association of it to any aspect of a security objective concerned with SCD/SVD generation facilitates merging of OT.Init with OT.EMSEC_Design and OT.SCD_Secrecy.

Threats that a SSCD Type 3 must counter are stated in the following. Relationship of objectives and security environment is given in the security objective rationale of the SSCD PP.

T.Hack_Phys	Physical attacks through the TOE interfaces
T.SCD_Divulg	Storing, copying, and releasing of the signature-creation data
T.SCD_Derive	Derive the signature-creation data
T.Sig_Forgery	Forgery of the electronic signature
T.Sig_Repud	Repudiation of signatures
T.SVD_Forgery	Forgery of the signature-verification data
T.DTBS_Forgery	Forgery of the DTBS-representation
T.SigF_Misuse	Misuse of the signature-creation function of the TOE

D. Modeling Common Criteria Security Targets

Schemes for developing trustworthy products using the Common Criteria as a framework have been proposed (e.g. [10], [11], [15], [16]) but none of them provides concrete tools for developers to meet the Common Criteria requirements for the content and expression of assurance artifacts. A technique is proposed herein to model the security environment and security objectives of a TOE as specially interpreted UML use case diagrams. The technique allows expression of ST artifacts using semiformal style and requires a systematic treatment of protected assets and subjects interacting with the system as part of a ST. The technique also assists, as is demonstrated in this paper, in analyzing Protection Profiles and Security Targets as developer is directed towards clear and consistent diagrams.

III. Modeling technique

A. Diagram specification

Security environment is modeled as a UML use case diagram. Use case diagrams are used for modeling system functionality at a very high level of abstraction and for defining the system boundaries and external systems and actors that area active within the context. The following interpretations of use case diagram artifacts enable modeling of the security environment:

1) Assets are represented as actors. Each asset 'acts' in the context of a security objective that concerns that asset. It is recommended to follow a specific notation of a prefix and name in the naming of actors representing assets. Each asset can be included in the diagram by name only but the diagram must be supported by a data dictionary where full descriptions of assets are given.
2) Subjects are represented as actors as is common practice in use case diagrams. It is recommended to follow a specific notation of a prefix and name in the naming of actors representing subjects.
3) Devices that constitute the IT-environment of the TOE are represented as actors. This is as in common use case diagrams. It is recommended that each external device is uniquely identified with a prefix that indicates that the particular actor is an external IT-device and a name.
4) Each assumption and OSP is represented as an actor. It is recommended that each item is uniquely identified with a prefix that indicates that the particular actor is an assumption or OSP and a unique name. Actors representing assumptions are not associated to those use cases that represent security objectives for the TOE but to those that represent security objectives for the environment. As devices constituting the IT-environment are modeled as actors, the assumptions and policies can not directly relate to external IT devices. Instead, they are associated to security objectives for the environment that are also associated to actors representing IT-environment but are outside the system (i.e. TOE) boundary. This is an important restriction as each security objective for the TOE must be addressed by security functional requirements with a possible support of OSP's and only security objectives for the environment of the TOE can be addressed (fully or partially) by assumptions.
5) Use cases associated to actors representing assets represent generic security objectives for the TOE or for the IT-environment of the TOE. It is recommended that the objectives are uniquely identified with a prefix that indicates whether it is an objective for the TOE or for the environment. Security objectives need not be fully defined as definitions can be derived from the model depending on which aspects of the objective are included. Generation of the definition from the design of security objectives shall be demonstrated in Sect. III-B. Generic security objectives should not be associated to actors representing subjects as they are further refined with aspects of security objectives accessible to subjects.
6) Includes-relationship illustrates the refinement of a security objective into aspects of the generic objective that are preserved by the TOE or IT-environment of the TOE. The naming, in order to facilitate derivation of meaningful definitions of security objectives, should be possible actions taken on the asset subject to generic security objective. Subjects and external IT-devices are associated to aspects of generic objectives as relevant.
7) Extension points are used to define the threats for each aspect of the security objective. Threat naming should follow the uniqueness conventions. While the generic security objectives for the TOE should not contain extensions points to facilitate configuration of that generic objective with different aspects depending on the intended use, each refinement of a generic security objective should be self-contained.
8) System boundary is expressed using the common use case notation as a rectangle separating the use cases from actors. In this case, the use cases representing security objectives for the TOE are also separated from use cases that represent security objectives for the IT environment of the TOE.

B. Generation of evaluation artifacts

Common Criteria requires that evaluation artifacts are unambiguously defined and the relationships between artifacts demonstrated. While definitions are given in the security environment specification, relationship tracings are usually given in the rationale. In this case, the rationales of interests are the security objective rationale covered by the following assurance elements:

1) *ASE_OBJ.1.2D* The developer shall provide the security objectives rationale.
2) *ASE_OBJ.1.2C* The security objectives for the TOE shall be clearly stated and traced back to aspects of the identified threats to be countered by the TOE and/or organizational security policies to be met by the TOE.
3) *ASE_OBJ.1.3C* The security objectives for the environment shall be clearly stated and traced back to aspects of identified threats not completely countered by the TOE and/or organizational security policies or assumptions not completely met by the TOE.

4) *ASE_OBJ.1.4C* The security objectives rationale shall demonstrate that the stated security objectives are suitable to counter the identified threats to security.
5) *ASE_OBJ.1.5C* The security objectives rationale shall demonstrate that the stated security objectives are suitable to cover all of the identified organizational security policies and assumptions.

In practice this means that the relationship of security objectives to threats, assumptions and OSP's must be demonstrated. A cross-tabulation is a common way to demonstrate the relationship and, after demonstrating the derivation of a definition for security objectives we shall demonstrate ho the use case for the security environment can be used for deriving the tracings for the rationale.

1) Definitions of security objectives: Assuming that exists a suitable definitions for assets in the data dictionary, definitions for security objective can be derived by concatenating the asset definition with the names of all the included aspects (named in a suitable manner) of the security objective (and the aspects related to those aspects) included in the diagram separated by suitable key words.

2) Derivation of tracings for rationale: One of the advantages of the proposed technique is that it facilitates semi-formal modeling of the definition of security. While the formality of assurance evidence for evaluated products increases from informal to semi-formal and formal once the evaluation assurance level increases, the assurance requirements for ST remain unchanged independently of the claimed evaluation assurance level. While bringing the modeling aspects to the security environment (and further to the entire ST) it is possible to subject ST to the same increase in rigor of modeling that the development assurance artifacts, namely the security policy model, high-level design and low-level design, as well as their correspondence evidence, and gain further assurance of appropriateness of design artifacts relative to the ST definitions. This is also likely to simplify the ST authoring. A significant portion of a ST consists of the rationale demonstrating appropriateness and correspondence of specifications.

While the semiformal model of security environment and security objectives acts as a tracing tool between security environment and security objectives, especially from security objectives to security environment, the inverse direction is not as obvious from the diagram (several threats, assumptions and policies may be relevant to more than one security objective, each security objective possibly being expressed in a separate diagram) so it may be, depending on the requirements of the national scheme, necessary to generate more unambiguous mappings from the security environment diagram. Such tracings are often expressed as a matrix of objectives and security environment artifacts. Such tracing matrix can be produced from the diagram in a straightforward manner:

1) Each security objective for the TOE and each security objective for the IT environment of the TOE is included as a row in a matrix.
2) Each threat, assumption and OSP is included in the column of the matrix.
3) For each aspect of each security objective for the TOE, an indication is included in the corresponding node in the matrix is the threat is relevant to that aspect.
4) For each security objective for the IT environment of the TOE, an indication is included in the corresponding node if that threat, assumption or OSP is relevant to that objective.

C. A Modeling exercise

OT.EMSEC_Design (Fig. 2) must be preserved during generation of SCD/SVD pairs (ASP.KEYGEN) and during computation of digital signatures (ASP.SIGN). Since it appears that asset AST.ES is related to OT.EMSEC_Design, aspect ASP.LEGAL ("preserving the legal validity of electronic signatures") is included. This helps to differentiate threats against technical device (T.Hack_Phys and T.Sig_Forgery) from threats against the PKI infrastructure (T.Sig_Repud). Asset AST.EMSEC_Data is created to represent the electromagnetic emanations generated during computation. The emanations must be such that no information about the AST.SCD can be deduced by recording and analyzing them. Radiation, however, is not a property of the key but the computation device and, while the inability of an attacker to correlate emanations protects SCD, the actual asset is non-correlation of electromagnetic emanations.

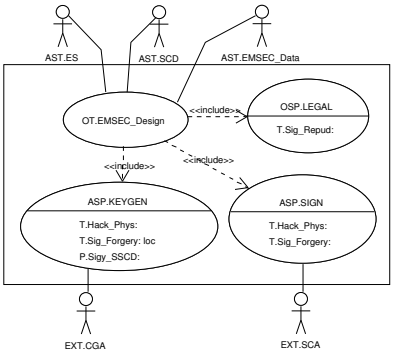


Fig. 2. OT.EMSEC_Design

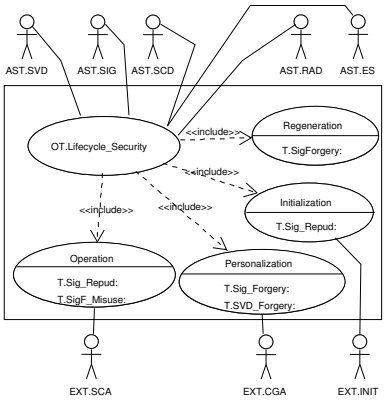


Fig. 3. OT.Lifecycle-Security

OT.Lifecycle_Security (Fig. 3) concerns with AST.SCD but security objectives rationale indicates that only T.SigForgery and T.SigRepud are relevant. On the other hand, the objective by definition concerns with initialization, personalization, operation and re-generation of the TOE. Initialization concerns with downloading the TOE to the smart card and configuring the file structure or equivalent. Personalization concerns with generation of SVD/SCD pair and RAD and with export of SVD to the CGA. Operational phase concerns with all aspects of signature function, and regeneration with secure destruction of SCD and generation of a new SVD/SCD pair. An additional threat concerned with residual information could be considered but it is assumed that T.Sig_Forgery definition is extended to cover reconstruction of AST.SCD from any residual information. T.SigRepud may be considered relevant to aspect "initialization" as inappropriate initialization may facilitate repudiation of signatures. Remote trusted IT product EXT.INIT representing the facility initializing the TOE (prior to personalization) is included to align the environment of the TOE with the security objective definition.

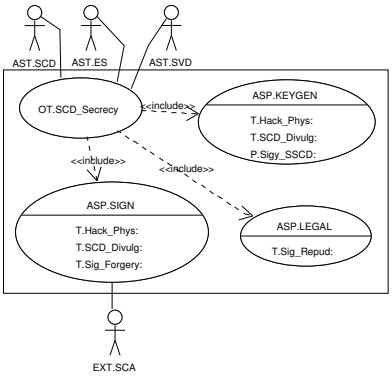


Fig. 4. OT.SCD_Secrecy

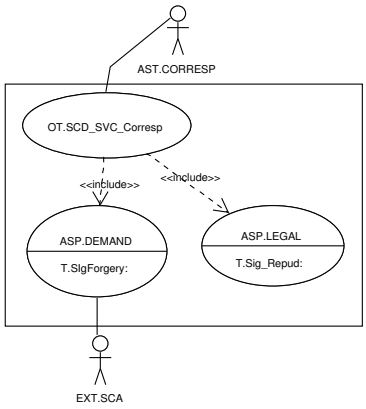


Fig. 5. OT.SCD_SVD_Corresp

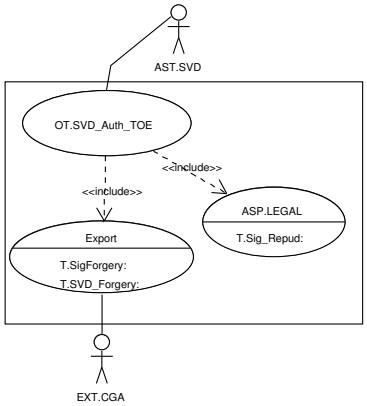


Fig. 6. OT.SVD_Auth_TOE

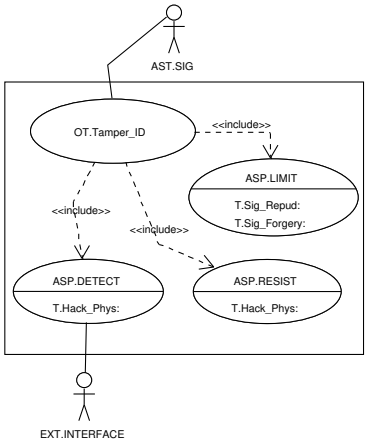


Fig. 7. OT.Tamper_ID

Relevance of AST.SVD, AST.RAD, AST.SIG and AST.ES to OT.Lifecycle_Security also implies relevance of T.SigF_Misuse and T.SVD_Forgery. Yet, initialization is unclear as no threat is directly relevant. Threat T.SIG_Integrity could be defined to indicate an attempt to modify AST.SIG prior to operational life-cycle phase. Amending the PP with TOE specific operations is, however, left to the developer. Initialization is also concern of assurance classes *ADO: Delivery and Operation* and *ALC: Life-cycle security* that are highly TOE and development environment specific. ST must define security objectives additional to those defined in the PP to counter, by suitable SFRs, threats specific to the TOE life-cycle.

OT.SCD_Secrecy (Fig. 4) concerns with AST.SCD but it is not clear whether it is relevant for AST.SCD used for signature creation during storage or during signature creation only. Security objective rationale indicates that T.HackPhys, T.SCD_Divulg, T.Sig_Forgery and T.Sig_Repud are relevant. Since T.Sig_Repud concerns with AST.ES and T.SCD_Divulg refers to release of SCD during generation, storage and use, AST.ES and AST.SVD are relevant. The latter because if SCD/SVD is of low quality, SCD may be divulged from SVD. OT.SCD_Secrecy must, to cover T.SCD_Divulg, be preserved during generation and use of keys. A strict interpretation of OT.SCD_Secrecy allows omitting the possibility of violation during key personalization and T.SVD_Forgery shall not need to be considered relevant. This also implies that ASP.KEYGEN is not associated to the CGA which may have implications on the trustworthiness of key generation as SVD is not passed to CGA immediately after generation. This may also have legal implications in some member states. OT.EMSEC_Design and OT.SCD_Secrecy appear identical but differ by the association to different assets.

OT.SCD_SVD_Corresp (Fig. 5) requires specification of AST.CORRESP (”Correspondence of SCD and SVD”). Demonstration of correspondence does not concern with AST.SCD or AST.SVD but with the correspondence of those which can not be easily encoded into specifications of AST.SCD and AST.SVD. Correspondence must be verified on demand but the security objective rationale indicates that T.Sig_Forgery and T.Sig_Repud are relevant. This implies that aspect ASP.LEGAL is relevant but verification on demand must be interpreted by developer. SSCD PP [14, Ch.5.1.5] defines security functional requirement *FPT_AMT.1: Abstract Machine Testing* that is relevant but does not provide further details of intention. ST developer must implement a selection of when the tests, some of which could verify the correspondence, are executed, for example upon explicit request by cardholder or by automated verification upon signature computation or SCD generation. This is highly implementation dependent and can not be encoded into the PP in a more general manner. Therefore, aspect ASP.DEMAND is included in the diagram but actual ST specifications may be significantly different.

Interpretation of OT.SVD_Auth_TOE (Fig. 6) is straightforward. Relevance of T.Sig_Repud, however, requires inclusion of ASP.LEGAL but T.Sig_Forgery can be considered relevant to aspect ”Export”. OT.Tamper_ID (Fig. 7) can also be interpreted in a straightforward manner. Aspects ASP.DETECT and ASP.LIMIT are derived from the definition. Relevance of T.Sig_Repud is unclear but can be considered countered if potential attacks can be contained to not violate AST.SIG. OT.Tamper_Resistance is similar OT.Tamper_ID

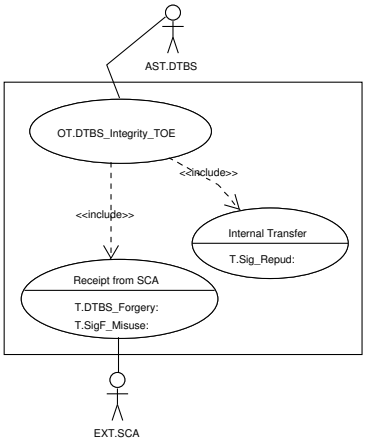


Fig. 8. OT.DTBS_Integrity

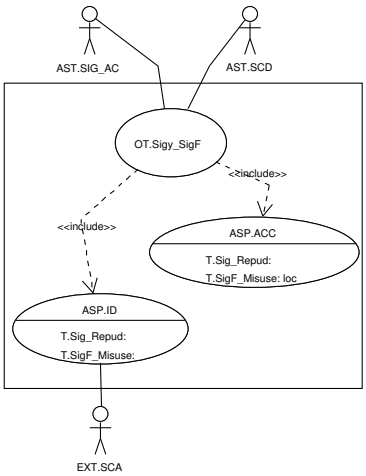


Fig. 9. OT.Sigy_SigF

and is incorporated as an aspect of OT.Tamper_ID.

Only P.Sigy_SSCD is relevant to OT.INIT, none of the threats, so it can be addressed by security objectives including aspect SP.KEYGEN and ”Initialization”. OT.SCD_Unique can be incorporated into objectives that concern with the generation of SCD/SVD pairs and incorporates aspect ASP.LEGAL. Threats relevant to ASP.LEGAL can be amended with T.SCD_Derive to indicate that the uniqueness of keys and inability to derive SCD from SVD as is a requirement for the legal validity of electronic signatures.

OT.DTBS_Integrity_TOE (Fig. 8) concerns protection of AST.DTBS during receipt from SCA and during any internal TOE transfer. OT.Sigy_SigF (Fig. 9) requires several interpretations. Asset AST.SIG_AC (”Legitimacy of access to signature function”) is created to differentiate access to from the integrity of signature function (AST.SIG). Legitimacy of access and confidentiality of SCD (AST.SCD) must be preserved during cardholder verification (ASP.ID) and granting access to SCD (ASP.ACC). Both threats relevant to OT.Sigy_SigF are relevant to both aspects.

OT.Sig_Secure can be considered as incorporated into the combination of security objectives that concern with the quality of cryptographic keys and the secrecy of keys during key generation and use for signature generation. It is questionable whether a separate security objective is required to further highlight the trustworthiness of signature function and parameters for signature computation. Trustworthiness of algorithms and parameters is assured by implementing

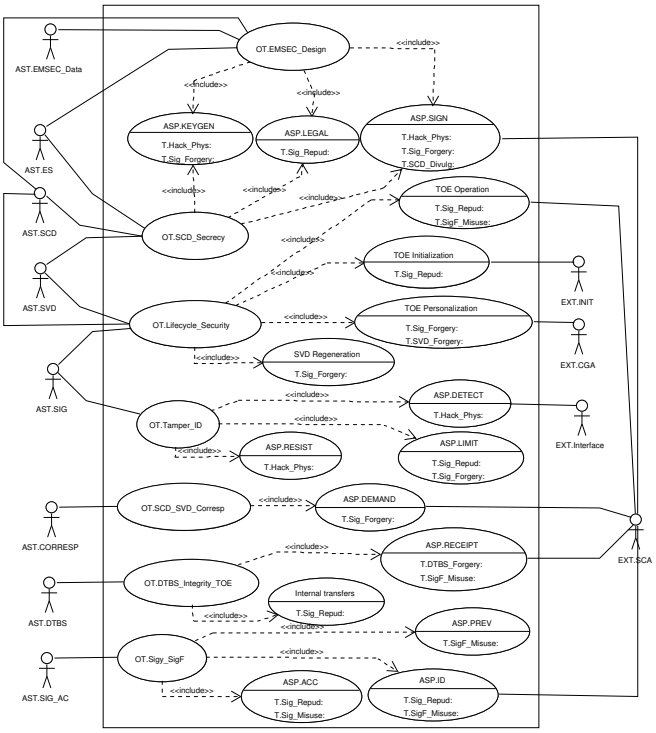


Fig. 10. Proposed security objectives of a SSCD

an appropriate subset of those defined in [9] and is hard to be coded into a PP. Alternatively, OT.Sig_Secure could be defined as a policy mandating use of approved algorithms and parameters when computing signatures and to associate that policy to all security objectives including ASP.SIGN.

IV. Findings on SSCD PP

Figure 10 illustrates a combined diagram for all security objectives of a SSCD Type 3, revised based on the above analysis. An additional aspect ASP.PREV is added to OT.Sigy_SigF to further highlight the stringency of access controls. To preserve clarity of the diagrams, acronyms are used for the naming of assets and aspects that can not be described with one or two words (Fig. 11).

Policies and assumptions (the latter of which are only relevant to security objectives for the environment excluded from the analysis) are omitted from the diagram illustrating the proposed changes to the SSCD PP to focus on the objectives and threats. Proposed changes also require changes in the security objectives to security environment tracings as part of the security objectives rationale. As no new threats are introduced or existing ones omitted, no major changes are expected to the security functional requirements of the PP but the relevant of different SFRs to security objectives and the related suitability claims change significantly.

The diagram can be used for generating also tracings of security objectives to the security environment as part of the security objectives rationale of a ST or PP. Statements of suitability of security objectives, also required from the security objectives rationale, can be generated from the diagram given an additional data dictionary defining threats. Definitions of revised security objectives are, based on the above analysis of the SSCD PP, argued to be more precise and accurate, and less prone to duplicates than the original ones. The following definitions are derived for revised security objectives:

1) *OT.EMSEC_Design* Confidentiality and integrity of the Signature Creation Data (SCD), unforgeability of Electronic Sig-

Assets:

AST.DTBS	Integrity of Data To Be Signed (DTBS) and representation thereof
AST.ES	Unforgeability of Electronic Signatures (ES)
AST.EMSEC_Data	Non-correlation of electromagnetic radiation
AST.SCD	Confidentiality and integrity of the Signature Creation Data (SCD)
AST.SIG	Authenticity and integrity of signature computation function
AST.SVD	Authenticity of Signature Verification Data
AST.SIG_AC	Legitimacy of access to signature function
AST.CORRESP	Correspondence of SCD and SVD

Aspects of security objectives:

ASP.LEGAL	Preserving the legal validity of electronic signatures
ASP.SIGN	Computation of digital signatures
ASP.KEYGEN	Generation of SCD/SVD pairs
ASP.DEMAND	Demand by an authorized party
ASP.RECEIPT	Receipt of DTBS from SCA
ASP.ID	Cardholder ID verification
ASP.ACC	Granting access to SCD
ASP.PREV	Preventing access to SCD from unauthorized parties
ASP.DETECT	Detecting an attempted physical attack
ASP.RESIST	Resisting an attempted physical attack
ASP.LIMIT	Limiting the exposure of sensitive data during an attempted physical attack

Fig. 11. Data dictionary for the proposed security objectives for a SSCD PP

natures (ES), and non-correlation of electromagnetic radiation are protected during preserving the legal validity of electronic signatures, computation of digital signatures and generation of SCD/SVD pairs.

2) *OT.SCD_Secrecy* Confidentiality and integrity of the Signature Creation Data (SCD), unforgeability of Electronic Signatures (ES) and authenticity of Signature Verification Data (SVD) are protected during preserving the legal validity of electronic signatures, computation of digital signatures and generation of SCD/SVD pairs.
3) *OT.Lifecycle_Security* Authenticity of Signature Verification Data (SVD) and authenticity and integrity of signature computation function are protected during TOE Initialization, TOE Personalization, TOE Operation and SVD Regeneration.
4) *OT.Tamper_ID* Authenticity and integrity of signature computation function is protected during Detecting an attempted physical attack, Resisting an attempted physical attack and Limiting the exposure of sensitive data during an attempted physical attack.
5) *OT.SCD_SVD_Corresp* Correspondence of SCD and SVD is protected during demand by an authorized party.
6) *OT.DTBS_Integrity_TOE* Integrity of Data To Be Signed (DTBS) and representation thereof is protected during Receipt of DTBS from SCA and Internal transfers.
7) *OT.Sigy_SigF* Legitimacy of access to signature function is protected during cardholder ID verification, granting access to SCD, and preventing access to SCD from unauthorized parties.

Main benefits of the diagrammatic modeling of security objectives is, not surprisingly, gained by clarifying the most complex security objectives, OT.EMSEC_Design, OT.Lifecycle_Security, and OT.SCD_Secrecy. Some duplication of security objectives were also removed and some irregularities in the scope of objectives made uniform by combining OT.Tamper_ID and OT.Tamper_Resistance. Several of the ambiguities with the SSCD PP resulted from lack of clarity in asset definitions. This suggests that a more stringent treatment of assets in Protection Profiles and Security Targets might assist developers and produce further assurance on the appropriateness of PP and ST specifications of security environment.

V. Findings on Common Criteria

Ambiguities in the SSCD PP are partially caused by inappropriate treatment of assets. Security environment of a PP or ST requires identification of assets but does not mandate on analysis on the suitability of security objectives to protect those assets. Threats may be stated with respect to assets they may violate but such treatment is not mandated by Common Criteria. Consequently, the findings from the analysis suggest enhancement of the Common Criteria treatment of assets.

Assurance classes *APE:Protection Profile Evaluation* and *ASE: Security Target of Evaluation* can not be extended as simply as assurance classes constituting the evaluation assurance levels. Conformance claims that allow extended and augmented EALs ([1, Ch.5.4]) are only relevant to the evaluation of TOE, not to the evaluation of PPs or STs. Consequently, there is no formal support in the Common Criteria for extending PP and ST evaluation. Developer may express and justify the extensions to the treatment of assets in the introduction of the ST but evaluators are not required to include those extensions in the evaluation reports and verdicts. Therefore, extending the treatment of assets requires also extending the philosophy of PP and ST evaluation. This is subject to approval from the CC International Management Board (CCIMB) but suggestion for extensions are proposed herein.

Extensions to assurance family are identical on Protection Profiles and Security Targets. However, formal descriptions are only given with respect to assurance class *ASE: Security Target Evaluation* to preserve compactness of the paper. Identical families to *ASE_DES: TOE Description*, *ASE_ENV: Security Environment* and *ASE_OBJ: Security Objectives* exist also in assurance class *APE: Protection Profile Evaluation*.

Proposed extensions are formally defined in Fig. 12. Naming follows the CC syntax and numbering allows adding the elements hierarchically to relevant components. This allows definition of assurance components *ASE_DES.2*. *ASE_ENV.2* and *ASE_OBJ.2* hierarchical to current components, i.e. component .2 includes all requirements from component .1 enhanced by the proposed extensions.

Developer action element *ASE_DES.2.2D* requires that the developer as part of ST introduction (PP introduction correspondingly) states the extensions followed in the ST evaluation. A corresponding requirement for the content and presentation of evidence elements explicitly states that those extensions are to be followed by the ST. A statement could be of form ”This ST follows the requirements for a ST stated in assurance class *ASE: Security Target Evaluation* extended with assurance components ASE_DES.2, ASE_ENV.2 and ASE_OBJ.2 defined as above”. This mandates no additional evaluator action elements.

An extension to the statement of security environment *ASE_ENV.2.2D* mandates inclusion of an explicit statement of protected assets. Corresponding requirements for content and presentation further explicate that requirement. Assets must be identified and protected security properties defined. Asset descriptions then take the form ”Confidentiality and integrity of SCD” where ”SCD” identifies the asset and ”Confidentiality and integrity” defines the security properties preserved. Evaluator action element *ASE_ENV.2.3E* is added that assurance of relevance of the assets to the ST.

There is no need to create additional developer action elements to assurance family *ASE_OBJ* but the statements for content and

Developer action elements:

ASE_DES.2.2D	The developer shall provide a statement of ASE extensions
ASE_ENV.2.2D	The developer shall provide a statement of assets as part of the ST

Content and presentation of evidence elements:

ASE_DES.2.2C	The statement of ASE extensions shall identify extensions to assurance class ASE followed by the Security Target
ASE_ENV.2.4C	The statement of assets shall identify any asset protected by the TOE
ASE_ENV.2.5C	The statement of assets shall state each relevant security property of protected assets
ASE_OBJ.2.6C	The security objectives for the TOE shall be traced back to assets covered by those security objectives
ASE_OBJ.2.7C	The security objectives for the environment shall be traced back to assets covered by those security objectives
ASE_OBJ.2.8C	The security objectives rationale shall demonstrate that the stated security objectives are suitable to protect the identified assets.
ASE_OBJ.2.9C	The security objectives rationale shall demonstrate that the stated security objectives are suitable to cover all security properties of assets

Evaluator action elements

ASE_ENV.1.3E	The evaluator shall confirm that assets are relevant to the ST

Fig. 12. Proposed extensions to Common Criteria

presentation of security objectives rationale is extended to indicate the relevance of stated assets to the security objectives for the TOE and environment of the TOE. Evaluator action elements for security objectives are also generic enough to cover the extensions to the rationale, no new ones are introduced.

VI. Conclusions and future work

Three generic principles for the treatment of security objectives are proposed and their Common Criteria interpretation given. While the principles remain generic, detailed and specific example has been used to illustrate the complexities ignorance of those principles introduces in the security specifications. The aim of the principles is to assist developers of IT security products in the construction of security specifications that are not only syntactically correct with respect to Common Criteria requirements but also capture the security problem of the product accurately and with uniform abstraction.

As the paper concerns of general principles, avenues for further research are numerous. In addition to analyzing further applications and PP's, various metrics shall be developed to quantify security specifications and to aim at identifying the complexity of PP's and ST's. The modeling technique used and integration with further high assurance design artifacts are also investigated to aim at a comprehensive modeling approach throughout the IT security product development life-cycle.

References

[1] Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model. ISO/IEC 15408-1, 1999.

[2] Information technology – Security techniques – Evaluation criteria for IT security – Part 2: Security Functional Requirements. ISO/IEC 15408-2, 1999.

[3] Information technology – Security techniques – Evaluation criteria for IT security – Part 3: Security Assurance Requirements. ISO/IEC 15408-3, 1999.

[4] Arrangement on the Recognition of Common Criteria Certificates in the field of Information Technology Security. http://niap.nist.gov/cc-scheme/ccra.pdf, May 2000.

[5] Common Evaluation Methodology for Information Technology Security CEM–97/017 Part 1 : Introduction and general model, Version 0.6, January 11 1997.

[6] Common Methodology for Information Technology Security Evaluation CEM–99/045 Part 2: Evaluation Methodology Version 1.0, August 1999.

[7] Commimssion Decision of 14 July 2003 on the publication of reference numbers of generally recognised standards for electronic signature products in accordance with directive 1999/93/ec of the european parliament and of the council. Official Journal of the European Union, L 175/45, July 15 2003.

[8] Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community Framework for electronic Signatures, December 1999.

[9] Electronic Signatures and Infrastructures (ESI); Algorithms and Parameters for Secure Electronic Signatures. ETSI Special Report SR 002 176, Version 1.1.1, March 2003.

[10] R. Kruger and J. Eloff. A common criteria framework for the evaluation of information technology systems security. In L. Yngstrom and J. Carlsen, editors, *Information Security in Research and Business – Proceedings of the IFIP TC11 13th International Conference on Information Security (SEC'97)*, pages 197–192. Chapmann & Hall, 1997.

[11] J. Leiwo. A mechanism for deriving specifications of security functions in the CC framework. In T. Bench-Capon, G. Soda, and A. M. Tjoa, editors, *Database and Expert Systems Applications, 10th International Conference DEXA'99*, number 1677 in Lecture Notes in Computer Science, pages 416–425. Springer–Verlag, 1999.

[12] Mutual Recognition Agreement of Information Technology Security Evaluation Certificates - Version 2.0. Management Committee of Agreement Group, Available at http://www.cesg.gov.uk/site/iacs/itsec/media/formal-docs/MRA99.pdf, April 1999.

[13] P. G. Neumann. Principled Assuredly Trustworthy Composable Architectures: Final Report. Available at http://www.csl.sri.com/neumann/chats4.pdf, SRI International, Menlo Park, CA, USA, 2003.

[14] Secure Signature–Creation Devices 'EAL4+'. CEN Workshop Agreement CWA14169, March 2002.

[15] R. Von Solms and H. Van De Haar. From trusted information security controls to a trusted information security environment. In S. Qing and J. H. Eloff, editors, *Information Security for Global Information Infrastructures – Proceedings of the IFIP TC11 16th Annual Working Conference on Information Security*, pages 29–36. Kluwer Academic Publishers, 2000.

[16] M. Wetterling, G. Wimmel, and A. Wisspeintner. Secure systems development based on the Common Criteria: the PalME project. In M. L. Soffa and W. Griswold, editors, *Proceedings of the tenth ACM SIGSOFT Symposium on Foundations of Software Engineering*, pages 129–138. ACM Press, 2002.

A Security analysis of Wi-Fi Protected Access™

Sandro Grech **Jani Nikkanen**

Dept. of Computer Science and Engineering
Helsinki University of Technology

{sandro.grech, jani.nikkanen}@hut.fi

***Abstract*—The convenience of WLAN (IEEE 802.11), combined with low associated investment costs, has led to a widespread adoption of the technology in both residential and commercial environments. Despite its success, the standard has also received a fair share of attention associated with its lack of robust security. In an attempt to overcome these security issues, IEEE is specifying a new generation of WLAN security, called IEEE 802.11i, which introduces a new type of wireless network known as Robust Security Network (RSN). In the meantime, due to the urgency associated with the current WLAN vulnerabilities, the industry has opted to roll-out WLAN implementations based on a draft version of the IEEE 802.11i standard, resulting in a wireless access mode known as Wi-Fi Protected Access (WPA). This paper presents results from a series of practical security attacks against a WLAN network implementing WPA.**

***Index Terms*—Communication system Security, Security evaluation, Wi-Fi Protected Access, Wireless LAN.**

I. Introduction

DUE to its wireless nature, IEEE 802.11 exhibits certain properties that give rise to vulnerabilities which are not exhibited in a fixed LAN environment. In order to protect against these vulnerabilities, a WLAN network requires the provision of access control and mutual authentication, together with strong privacy protection mechanisms. The current IEEE 802.11 standard [1] ineffectively attempts to meet these requirements through the Wired Equivalent Privacy (WEP) protocol. WEP includes two authentication modes, known as open authentication, and WEP authentication. When using open authentication, an access point always accepts the authentication request from a mobile station. However, in practice, many systems provide proprietary access control methods, the most popular being MAC address lists. WEP authentication, on the other hand, is based on a challenge-response mechanism using a shared key. WEP uses a stream cipher called RC4 to encrypt the data packets, using a concatenation of the secret key and an initialization vector as the RC4 key. An analysis of why WEP fails in all accounts to provide robust security is beyond the scope of this paper. Several publications cover the subject in detail (e.g. [6], [7] and [8]).

Due to the insufficient robustness provided by the IEEE 802.11 standard, most public WLAN systems deploy security at the higher layers of the protocol stack. HTTP redirection to a captive portal is most commonly used as an access control and authentication mechanism. Privacy is most commonly enabled by means of an IPsec tunnel between the mobile station and a VPN gateway.

In the meantime the IEEE standards Task Group on security (TGi) have been working on a standard solution for IEEE 802.11. This effort will result in a new standard, IEEE 802.11i, which specifies a new security architecture for WLAN, known as Robust Security Network (RSN). RSN builds on top of the IEEE 802.1X standard for port based network access control. The IEEE 802.1X standard [3], which was originally intended for wired IEEE 802 Local Area Networks, provides strong authentication, access control, and key management.

Due to the lengthy standardization process of IEEE 802.11i, coupled with pressure from the industry to provide prompt security solutions, the Wi-Fi Alliance, whose main intent is to certify interoperability of products based on IEEE 802.11, has adopted a subset of the IEEE 802.11i standard (based on a draft thereof [4]), into a specification called Wi-Fi Protected Access (WPA). The first commercial WPA products appeared on the market during late 2003.

II. Background

Wi-Fi Protected Access (WPA) includes two modes of operation. The basic mode (WPA with pre-shared key, or WPA-PSK, also known as WPA-personal) is most suited to residential or SOHO deployments, and does not require any back-end security infrastructure (Figure 1). In this mode of operation, access to the wireless network services is allowed only if the mobile device can prove the possession of a pre-shared key which matches that stored at the access point(s).

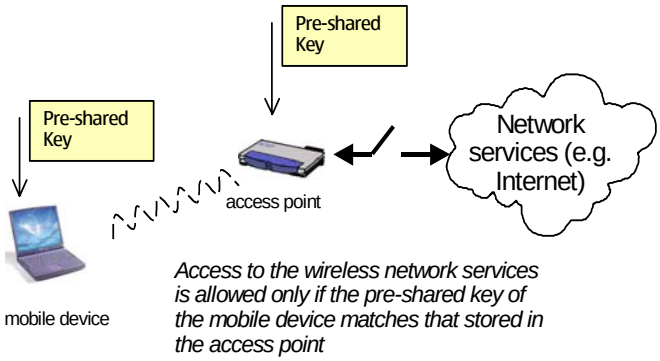


Figure 1 – WPA-PSK (WPA-personal) for residential/SOHO environment

A second mode of operation known as WPA-enterprise, is most suited to enterprise and other medium- to large-scale deployments, requires a back-end security infrastructure (Figure 2). In this mode of operation, security signaling between the mobile device and the authentication server is relayed through the access point. Access to the wireless network services is allowed only after the authentication server confirms that the mobile device has been successfully authenticated. WPA also replaces the weak WEP privacy protection mechanism with a new mechanism known as Temporal Key Integrity Protocol (TKIP).

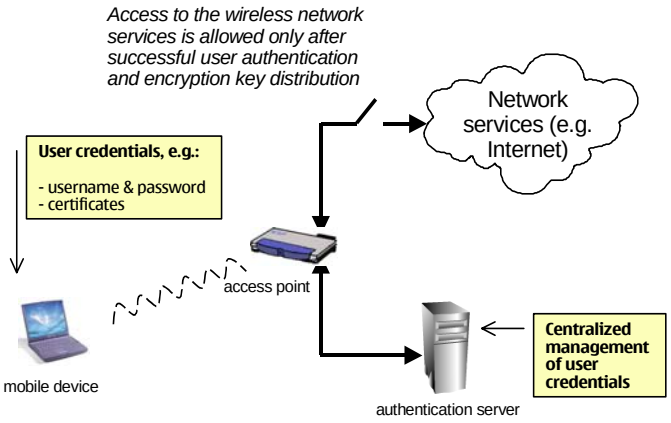


Figure 2 – WPA with centralized security management for enterprise environments

III. ATTACKS AND VULNERABILITIES

A. *Attacks against WPA communication integrity*

In [9], the authors disclose a man in the middle and session hijack attack against RSN. These attacks make use of forged 802.11 management frames and assume asymmetric authentication between the mobile station and access point. The validity of this assumption is strongly coupled to the EAP method used between the mobile device and the authentication server. By definition, if mechanisms supporting strong mutual authentication such as EAP-TLS [20] and EAP-AKA [17] are deployed, then the mentioned man in the middle attack will fail. The dominant WPA supplicants currently available on the market, however, also support tunneled authentication mechanisms such as EAP-TTLS [18] and PEAP [19] which have been shown to be susceptible to man in the middle attacks [21]. Consequently, until solutions are worked out, man in the middle attacks can only be avoided by using mechanisms such as EAP-TLS and EAP-AKA which require credentials in the form of digital subscriber certificates and SIM hardware, respectively.

WPA also requires an EAP method that generates keying material. This enables protection of the traffic between the mobile device and the access point. Consequently, the session hijack attack presented in [9] will be reduced to a denial of service attack. Denial of service attacks against WPA are covered in more detail in the following sections.

B. *Denial of Service attacks based on message spoofing*

One of the constraints that the improved WLAN security standards had to comply with, is reverse compatibility with implementations of the original IEEE 802.11 standard. In practice, this means that a WPA or RSN capable mobile station shall be able to interoperate with a non WPA or non RSN capable access point. As a consequence of this requirement, the WPA and RSN security frameworks build on top of the management frames that were introduced in the original release of IEEE 802.11. One notable characteristic of these management frames is that they are not secured. This gives rise to a series of opportunities for potential attackers for mounting denial of service attacks by forging these management frames.

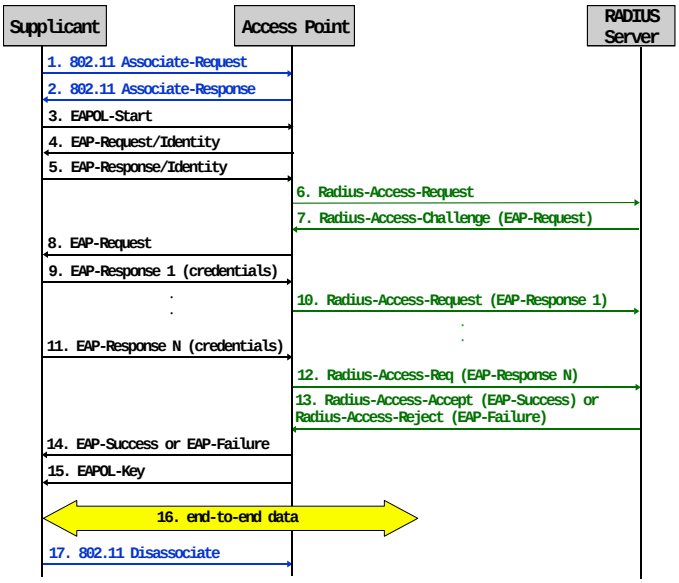


Figure 3 – An IEEE 802.1X authentication session in a WLAN environment

Figure 3 shows an IEEE 802.1X authentication session. The access point can in addition, at any stage, send an IEEE 802.11 *Deauthentication* management frame, in case it wants to revoke a successful authentication for a previously authenticated client.

There are various ways to launch a denial of service attack using these unsecured management frames:

i. flooding the access point with IEEE 802.11 *Associate-Request* or *EAPOL-Start* frames using random MAC addresses

ii. spoofing *EAP-Failure* messages using the access point's spoofed MAC address

iii. spoofing an IEEE 802.11 *Deauthentication* message from the access point's spoofed MAC address towards a selected victim or using the broadcast address.

The vulnerabilities caused by these attacks range from denial of service targeted to a specific device, to a denial of service against the whole network.

IV. PRACTICAL ATTACKS

In order to mount the attacks, we set up the WPA network depicted in Figure 5. The victim, access point/router and authentication server together form the target network. Mutual authentication between the mobile station and access point is based on EAP/TLS. The attacker consists of a laptop equipped with the *airjack* driver [10] on top of Red Hat Linux 8.0. A *prism 2* chipset based card is required for *airjack*. We are mainly interested in the *airjack* feature that allows us to send arbitrary IEEE 802.11 frames (including frames with spoofed MAC address) through the wireless adapter. We also availed of the *file2air* tool [12] which reads from a binary input file assumed to contain a valid IEEE 802.11 frame, and transmits the contents through the network adapter using the *airjack* driver.

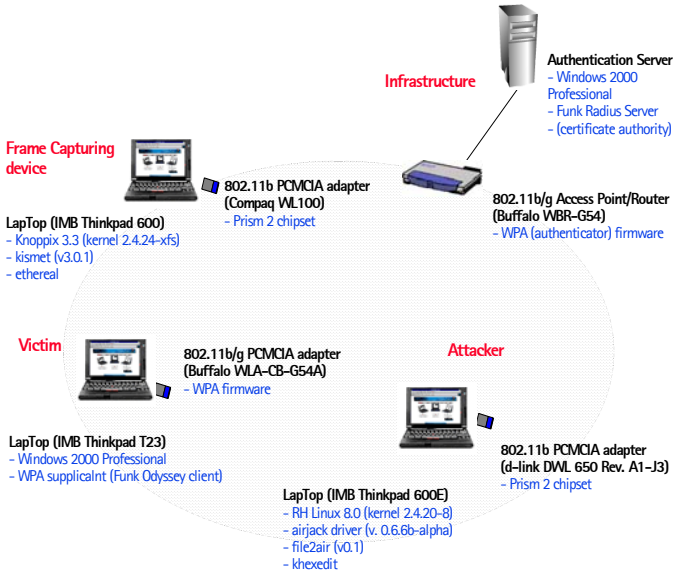


Figure 4 - Attack environment

In order to create the binary files we captured legitimate management frames from the WPA network using *kismet* [13]. *Kismet* is an IEEE 802.11 layer 2 wireless network detector, sniffer, and intrusion detection system. The relevant frames were extracted using *ethereal*, and later modified using the *khexedit* (Linux hex editor) tool.

In our experiments we were able to successfully mount denial of service attacks using forged IEEE 802.11 *Deauthentication* messages and IEEE 802.11 *Disassociation* messages. Figure 5 illustrates the IEEE 802.11 state machine. The frame classes are defined in [1]. Most notably, an IEEE 802.11 *Deauthentication* frame will cause a transition to state 1 in which the WLAN station cannot engage in any end-to-end data communication. An IEEE 802.11 *Disassociation* frame sent from a station residing in State 3 in the IEEE 802.11 state machine, will cause a transition to State 2. Once again, no end-to-end data communication can take place in this state.

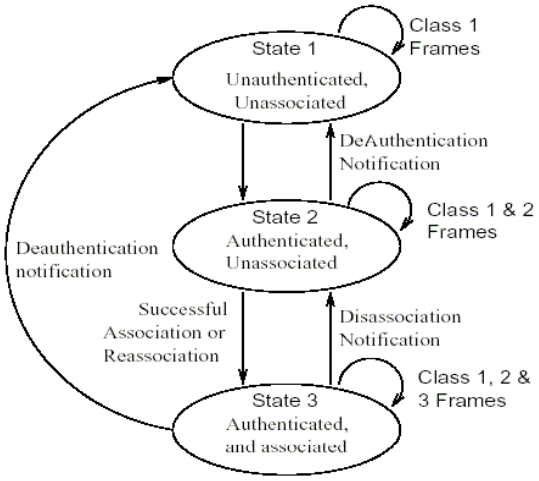


Figure 5 – IEEE 802.11 state machine [1]

Figure 6 shows a plot of the captured frames during the denial of service attack based on IEEE 802.11 *Deauthentication* frames as seen by the frame capturing device. The black bars indicate management frames, and the red bars indicate *Deauthentication* frames. The gap between 0.5 seconds and 19 seconds appears since *kismet* was apparently unable to detect the data frames after successful WPA authentication. These frames were encrypted with the new TKIP algorithm.

Shortly after the capture was started, the victim authenticates to the network. The first burst of frames corresponds to this procedure. The deauthentication flood was started after about 19 seconds, and lasted for about 3 seconds. During this period, the victim cannot engage in any end-to-end communication. After the flood was ceased, the client successfully reconnects to the network. This is shown by the second burst of black bars in the figure. The client in our setup manages to reconnect to the access point fairly quickly after

the attack is ceased, since the client is continuously and periodically attempting to reconnect.

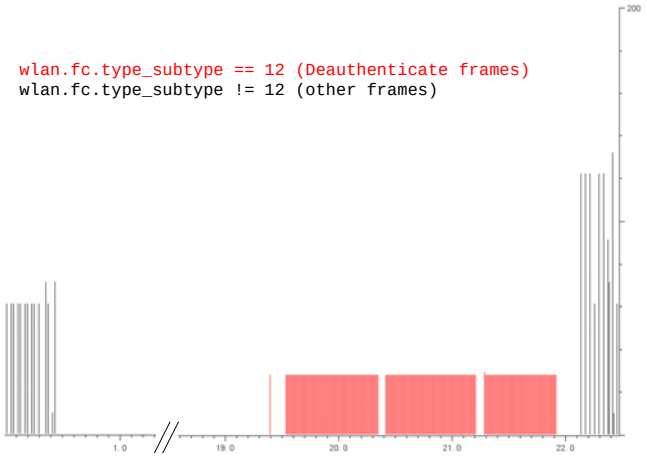


Figure 6 - transmitted bytes vs. test time

We also attempted to cause a denial of service by flooding the access point with IEEE 802.11 *Associate-Request* messages. In our setup we were able to inject 10 *Associate - Request* messages per second. Despite our attempted denial of service attack, we were unable to experience any service disruption and the legitimate client was able to successfully authenticate itself with the access point after the flood persisted for several seconds. We conclude that this failure is attributed to the fact that we attempted to mount the attack from a relatively slow (400 Mhz, Pentium II) machine, by today's standards.

Similarly, spoofed *EAP-Failure* messages failed to disrupt communication in our setup. We conclude that this is attributed to an attack synchronization issue. In order to be effective, the *EAP-Failure* message needs to be received by the supplicant after message 11 has been sent, but before message 14 is received (Figure 3). Despite the fact that we generated a continuous flood of *EAP-Failure* messages (approximately 10 messages per second), we were unable to succeed in this synchronization. The probability that this attack succeeds can be calculated as follows:

$$p = \frac{g}{f} \quad , \quad if \ \frac{g}{f} \le 1$$

$$p = 1 \quad , \quad otherwise$$

where p represents the probability for the attack to succeed, g represents the opportunity gap (time interval between message 11 and 14, in Figure 3) measured in seconds and f represents the frequency in Hz at which the forged *EAP-Failure* messages can be injected. In our case g was approximately 6ms and f was approximately 10Hz, giving us a probability of success of 0.06. In other words, on average, the victim needs to authenticate to the access point about 17 times, before the *EAP-Failure* has any impact on the victim's machine. More importantly, we have observed that our supplicant automatically periodically re-attempts to authenticate with the access point, following a failure. Consequently, even in the case that the forged *EAP-Failure* message is successfully delivered on time, its impact on the victim will be close to negligible.

V. Protection against the attacks

Protection against denial of service attacks based on forged IEEE 802.11 management frames is not so straightforward, since there is no standard way to detect forged IEEE 802.11 management frames from legitimate ones. Consequently, there are no standard ways to protect against these attacks. Intrusion detection systems with proprietary mechanisms for detecting attackers and discarding forged frames may be deployed. The mechanism presented in [14] for detecting forged frames may be applied, for example. This mechanism is based on the fact that without the ability to control the firmware functionality of wireless cards, and without the source code to develop custom firmware, an attacker does not have the ability to alter the value of the sequence number field in the 802.11 header. Forged frames can thus be detected by monitoring for a sudden change in sequence numbers.

VI. Conclusions

It is widely known that the security supported by the base IEEE 802.11 standard is inappropriate. The first commercial WPA products implementing enhanced security for IEEE 802.11 networks have recently started to appear on the market. We were able to successfully mount two variants of denial of service attacks. These vulnerabilities are inherited from the base WLAN standard. IEEE is aware of these vulnerabilities and has opted not to protect against them since the majority of the participants felt that a determined attacker can always resort to RF-based denial of service attacks (e.g. frequency jamming techniques).

The significance of denial of service attacks against WLAN is heightened especially as the wireless technology gains adoption into systems in such way which was unpredicted at the time that the technology was designed. As an example, major players from the telecommunications industry have formed a consortium which recently specified WLAN as a complementary access mechanism for GSM and GPRS services [22]. This includes services such as circuit-switched voice calls which have thus far supported an adequate level of security.

References

[1] Institute of Electrical and Electronics Engineers (IEEE). "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications". IEEE Std 802.11-1999, March 1999.

[2] Institute of Electrical and Electronics Engineers (IEEE): "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY): Higher Speed Physical Layer (PHY) Extension in the 2.4 GHz band. IEEE Standard 802.11b", September 1999.

[3] Institute of Electrical and Electronics Engineers (IEEE). “Standard for Port based Network Access Control”. IEEE Draft P802.1X/D11, March 2001.
[4] Institute of Electrical and Electronics Engineers (IEEE). "Draft supplement to standard for telecommunications and information exchange between systems - LAN/MAN specific requirements - Part 11: wireless Medium Access Control (MAC) and physical layer (PHY) specifications: specification for enhanced security". IEEE Draft 802.11i/D3, November 2002.
[5] Wi-Fi Alliance: “Wi-Fi Protected Access Documentation”, version 2.0, April 29, 2003.
[6] N. Borisov, I. Goldberg, and D. Wagner.: “Intercepting mobile communications: the insecurity of 802.11.” MOBICOM, 2001.
[7] J. Walker: “Unsafe at any key size: an analysis of the WEP Encapsulation”, Tech. Rep. 03628E. IEEE 802.11 committee, March 2000.
[8] W. A. Arbaugh, N. Shankar and J. Wang. “Your 802.11 Wireless Network has No Clothes”, In Proceedings of the first IEEE International Conference on Wireless LANs and Home Networks, December 2001.
[9] A. Mishra and W. A. Arbaugh: “An initial security analysis of the IEEE 802.1X standard”. Technical Report UMIACS-TR-2002-10.
[10] AirJack: http://802.11ninja.net/airjack/
[11] W. A. Arbaugh, N. Shankar and Y. J. Wan: “Your 802.11 wireless network has no clothes”. In Proceedings of the First International Conference on Wireless LANs and Home Networks (Singapore, 2001).
[12] file2air: http://home.jwu.edu/jwright/code/file2air-0.1.tar.bz2
[13] Kismet: http://www.kismetwireless.net
[14] J. Wright: “Detecting WLAN MAC address spoofing”, January 2003. http://home.jwu.edu/jwright/papers/wlan-mac-spoof.pdf
[15] L. Blunk, and J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", RFC 2284, March 1998.
[16] H. Haverinen and J. Salowey, "EAP SIM Authentication", draft-haverinen-pppext-eap-sim-13.txt, Internet draft (work in progress), October 2003.
[17] J. Arkko, H. Haverinen, "EAP AKA Authentication", draft-arkko-pppext-eap-aka-12.txt, Internet draft (work in progress), April 2004.
[18] P. Funk and S. Blake-Wilson, “EAP Tunneled TLS Authentication Protocol (EAPTTLS)”, draft-ietf-pppext-eap-ttls-04.txt, Internet draft (work in progress), April 2004.
[19] S. Josefsson, et al., "Protected EAP Protocol (PEAP)", draft-josefsson-pppext-eap-tls-eap-06.txt, Internet draft (work in progress), March 2003.
[20] B. Aboba, and D. Simon, "PPP EAP TLS Authentication Protocol", RFC 2716, October 1999.
[21] N. Asokan, V. Niemi, and K. Nyberg: "Man-in-the-middle in tunnelled authentication", In the Proceedings of the 11th International Workshop on Security Protocols, Cambridge, UK, April 2003.
[22] The Unlicenced Mobile Access (UMA) Consortium Specifications. Available: http://www.umatechnology.org.

Internet ID - Flexible Re-use of Mobile Phone Authentication Security for Service Access

Marko Schuba[1], Volker Gerstenberger[2], Paul Lahaije[3]

***Abstract*—Identity management systems like .NET Passport and the Liberty Alliance Project allow for different forms of authentication to be used. Although simple username and password schemes are very commonly used, these are not considered to be very secure. This paper describes the Internet ID approach for user authentication in identity management systems. The approach utilizes the huge existing base and high security of SIM smart cards in mobile phones to authenticate end users in Liberty Alliance based systems. The Internet ID approach is applicable to both the mobile and fixed Internet worlds, i.e. services can be accessed via a mobile terminal or via a PC. Authentication is, in both cases, done via the mobile phone and its SIM card, combining the security of the SIM with the "always on" capability and convenience of the mobile phone.**

***Index Terms*—Identity management, Single Sign-On, SIM, authentication,**

I. Introduction

Identity management, particularly the aspect of Single Sign-On (SSO), has gained a lot of attention in recent years. The main reason for this trend is the increasing number of Service Providers (SP) on the Internet that require users to establish login accounts before they grant access to their services. The account data typically include credentials – often in the form of a username and a password – that need to be presented when a user wants to log in to the SP site to use its services. As most people are unable or unwilling to handle more than a few different username/password combinations, they start either to use password management software on their PCs, to write passwords down, or to use the same username/password combinations for different accounts. This behavior makes it easier for attackers to get hold of the users' credentials and thus lowers the protection of the users' accounts in the Internet.

Identity management systems aim to reduce the number of passwords users have to handle through SSO: users do not have to authenticate themselves at different service providers (SP) separately, but can reuse a successful authentication from one SP to authenticate at other SPs. Typically the first authentication will be done towards a trusted party (often called an Identity Provider (IDP)), so that the user, in principle, only has to maintain a single set of credentials.

SSO not only improves the convenience for end users but may also improve security since users, only having to remember a single password, could select one that is hard to guess.

On the other hand there is also the risk that SSO makes it even easier for attackers: a badly protected password used for SSO might open up not only a single user account, but a large number of user accounts all over the Internet. Thus, the credentials that are to be used for SSO need to be particularly secure.

Commonly used authentication credentials are usually based on one, or a combination, of the following types: "something you know" (e.g. a password), "something you have" (e.g. a security token), and "something you are" (biometrics). In order to improve the security of the credentials used for SSO, the purely password-based approach should be extended.

The Internet ID concept demonstrates how the security of today's mobile phones (GSM authentication based on the Subscriber Identity Module (SIM)) can be re-used to improve the security of SSO authentication. We apply our approach to the Liberty Alliance protocols, as they allow IDPs to choose between different authentication methods. The flexibility of this approach enables users to utilize SIM security not only for mobile Internet services but also for services accessed via a PC. The authentication credentials can be exchanged over the cellular network or over the fixed Internet and Bluetooth.

The rest of the paper is structured as follows. Section II gives a brief introduction to two identity management systems: .NET Passport and the Liberty Alliance Project. Section III discusses the different authentication methods offered by the SIM card and the security advantages that the SIM provides. Section IV describes the Internet ID approach, which combines Liberty Alliance protocols with the authentication security of the SIM for universal service access. Finally, section V gives some concluding remarks and an outlook on future research.

II. Identity Management Systems

A tremendous number of identity management solutions exist in the market today and this section describes two of

[1] M. Schuba is with Ericsson Research, Corporate Unit in Aachen, Germany (phone: +49-2407-575-7782; fax: +49-2407-575--400; e-mail: marko.schuba@ericsson.com).

[2] V. Gerstenberger works for Giesecke & Devrient, Telecommunications in Munich, Germany (e-mail: volker.gerstenberger@de.gi-de.com).

[3] P. Lahaije works for Vodafone Group Research and Development in Maastricht, The Netherlands (e-mail: paul.lahaije@vodafone.com).

them in more detail: Microsoft .NET Passport and the Liberty Alliance Project. The reason for selecting these two is that both solutions provide a *framework* for identity management, whilst others are commercial products.

A. .NET Passport

Microsoft launched .NET Passport in 1999 [1, 2, 3]. It is a web-based authentication service, which provides SSO[1] to users. A participating user has one email address and password to log in to all .NET Passport enabled sites and services. .NET Passport stores personal information in a profile and this information can be automatically shared during SSO, thus allowing enabled sites to provide personalized services to the customer. A further option in the profile is the configuration of a .NET Passport wallet. In this case, attributes like credit card information and shipping addresses etc. are stored in the profile. To ensure higher security wallet information is stored separately from the base profile.

During registration of a .NET Passport account, a .NET Passport User ID (PUID) is generated that serves as a unique identifier for the account. The PUID, which is different from the user's email address, is used by SPs to identify the same person from one SSO session to the next.

When a user wants to access a protected service, the SP first checks the authentication status of the user. If authentication is required the user can click on a 'Sign-On' button that re-directs him/her to the .NET Passport Login server. The .NET Passport Login server checks the SP data (passed as parameters during re-direction) and, provided the SP is a partner, displays a sign-on page to the user. Upon positive authentication with email address and password, the .NET Passport Login server generates a set of three encrypted cookies and re-directs the user back to the SP. A Ticket cookie contains the PUID and a time stamp, whilst a Profile cookie stores profile data that the user has chosen to share with the SP. Both cookies are set for the SP's domain as well as for the .NET Passport domain and can therefore be decrypted with the SP's encryption key. The SP uses these two cookies to verify the authentication and to extract profile information. If the authentication was successful, access to the protected service is granted. The third cookie is used for the actual SSO process: this Visited Sites cookie contains the names of all sites to which the user has signed-on during the session. It is set only for the .NET Passport domain and is therefore only encrypted with a .NET Passport key. When an SP re-directs the user to the .NET Passport Login server, the server first checks for the existence of this cookie. Only if it does not exist or is no longer valid does the server ask the user for his/her credentials.

There is no direct server-to-server communication between the .NET Passport Login server and SPs servers for SSO. The information exchange occurs through the user's Internet browser using HTTP re-directs and cookies. However, .NET Passport does perform server-to-server communication periodically to update operational information about the locations of the .NET Passport servers.

In the future Microsoft plans to support additional authentication methods and security levels (e.g. Kerberos v5 authentication protocols), smart cards, and digital certificates within the .NET Passport framework.

B. Liberty Alliance Project

The Liberty Alliance Project (LAP), which was started in 2001, is a common effort of global companies and organizations to standardize a network identity management framework [4, 5, 6]. Its main objectives are to:

- Develop specifications that enable commercial and non-commercial organizations to protect consumer privacy;
- Provide an open SSO specification that includes federated authentication from multiple providers operating independently;
- Enable commercial and non-commercial organizations to control, maintain and enhance relationships with constituents;
- Create a network identity infrastructure that supports all current and emerging network access devices.

The first set of specifications developed by LAP focuses on the federation of accounts and SSO in a so-called Circle of Trust (CoT). A CoT is comprised of a set of SPs and one or more IDPs. The SPs have a business relationship with the IDP(s) that allows the IDP to authenticate users (who are called Principals in LAP) on their behalf. However, before this can happen, the Principal has to federate their respective SP account with their account at the IDP (in .NET Passport this is done implicitly without asking the user for consent). Once federated, the browser of a Principal, who tries to access a protected service at an SP, is re-directed to the IDP for authentication. If the authentication is successful the IDP issues a signed assertion that allows an SP to authenticate the Principal without further interaction, after his/her browser has been re-directed back to the SP. The Principal is now free to move between any (federated) SPs within the CoT in an SSO manner.

In order to protect the Principal's privacy the accounts used at the different SPs and IDPs can, and should, have different usernames. As LAP uses pseudonyms to map each username at an SP to a username at an IDP, the accounts of a Principal cannot easily be connected by any two SPs. The IDP however, which acts as a trusted intermediate, *could* collect more information about Principals and so it is important for Principals to select an IDP carefully, or use more than one IDP.

The LAP specifications do not standardize a single set of credentials to be used for authentication. Instead, LAP supports various so-called authentication contexts, which range from using a password sent over an unprotected link to Public Key Infrastructure (PKI) in combination with smart cards. It is up to the SP to decide if it accepts the

[1] .NET Passport actually uses the term Single Sign-In. For reasons of consistency, we use the term Single Sign-On throughout this paper.

authentication context offered by a particular IDP.

It should also be noted that the LAP specifications do not only support SSO, but also define a framework for permission-based sharing of user attributes among partners. This could enable services such as Profiles, Calendars, Personal Information Management, Wallets, Alerts etc.

III. Mobile Phone and SIM Security

The security design of mobile phones is usually more sophisticated than that of PCs or PDAs. For more than a decade GSM phones, for example, have used a smart card in the form of a SIM for authentication to GSM networks [7]. This means that a widely deployed base of smart cards exists today. Furthermore, these smart cards reside in an online smart card reader (i.e. the mobile terminal) that means that they can be accessed (almost) anywhere and anytime.

A smart card is a computing device, which is tamper-resistant to the extent that a malicious person can only obtain data stored or processed in the smart card by using very costly techniques (see e.g. [8]). Therefore smart cards are ideally suited for performing sensitive operations such as storing cryptographic keys and execution of cryptographic algorithms. For instance, if a user is to use a private key for public-key-cryptography operations (such as creating digital signatures), then it is often convenient, secure and practical to put this private key (and the operations relating to the use of the private key) on a smart card. It is perfectly possible for smart cards with a suitable coprocessor to produce 1024-bit RSA signatures in a commercially acceptable time.

The SIM is a key component for GSM (GPRS, EDGE) and UMTS authentication. (Note that although the SIM has been updated for UMTS, and is then referred to as a Universal SIM (USIM), we will continue to use the term SIM here to encompass both SIMs and USIMs.) For each subscription a shared secret key, K_i, is held in the SIM card and in the home operator's Authentication Centre (AuC). Authentication is based on a random challenge (Rand) and key-dependent expected response (XRES). The challenge and expected response are generated by the AuC and sent to the visited network. The visited network then sends the challenge over the air (via the mobile terminal) to the SIM, which uses the K_i to calculate a response (SRES). This SRES is returned, via the mobile terminal, to the visited network, which compares the SRES and XRES values. If they match, authentication has been successful. Simultaneously to the calculation of XRES and SRES a cipher key is generated, which can be used to secure the radio link. For UMTS this process is enhanced by the introduction of an authentication procedure that allows for mutual authentication of both the (U)SIM and the network.

Because the cryptographic computations required for authentication take place only on the SIM and in the AuC the key, K_i, never has to leave the SIM or the AuC. Indeed it is effectively infeasible for anyone to read the K_i from the SIM, or to deduce it by observing the I/O of the SIM. This is even the case if specific challenges (Rand) and responses (SRES) can be observed. (Weak authentication algorithms, or weak implementations of authentication algorithms, can reduce this security, but in general this assertion is true.) This is very important because it prevents SIMs from being cloned in the distribution chain, or by people who "borrow" another user's SIM, or (worse) by people sending Rand challenges over the air to the mobile. Hence it prevents customers from being able to refute their phone bills on the grounds that they may have been a victim of SIM cloning.

The GSM authentication process described above authenticates the SIM - i.e. the subscription - to the network. This in itself does not strictly authenticate the *subscriber*, since someone else might have stolen the phone and might use it to make calls. Therefore, the SIM provides a mechanism to lock the SIM to a particular customer by requiring the customer to enter a Card Holder Verification (CHV) PIN before the SIM can be used. (Note though that for normal GSM telephony use, the user may switch off this PIN request.)

In recent years the SIM's functionality has been extended, as follows:

- Additional protocols have been developed that use the SIM for more general authentication purposes (see e.g. [9, 10, 11]). To date this form of authentication still focuses on access network authentication (e.g. for WLAN) as opposed to service level authentication.
- In the context of WAP a SWIM card (SIM application plus a Wireless Identity Module (WIM) application [12] in the same smart card) can be used for two extra forms of authentication:

  1) WTLS authentication: here, the WIM and a network server can be mutually authenticated at the transport level. Depending on the version of WAP, the server end point is either the remote server or the WAP gateway. Authentication is based on public key algorithms (RSA or Elliptic Curve) using Diffie-Hellman key exchange and authentication certificates. Normally a URL is used to point to the location of WIM certificates, in order to save space in the smart card.

  2) User authentication: in this case the remote server authenticates the end user at the application level. This authentication can use two different public key algorithms: RSA digital signature or Elliptic Curve digital signature. The remote server sends a text to be signed and a challenge to the mobile phone using WMLScript. After the user has entered a mandatory signature PIN (the user cannot switch off this PIN requirement) the mobile phone calculates a hash of the text and sends it to the WIM application to calculate the digital signature. The digital signature and the URL of the user's signing certificate are then sent to the remote server for authentication. Note that the WMLScript command used for the signature (signText) only accepts text strings, so it is impossible to sign binary values.

Using SIM Application Toolkit (SAT), a large variety of authentication mechanisms can be implemented on a SIM card. For instance, public key authentication, shared secret, password etc. can all be implemented. As a consequence, SAT enables SIM cards to support multiple different authentication methods. SAT is also bearer independent and so any bearer can be used to exchange the authentication messages with the remote server. Currently SMS is the most commonly used technology.

Another feature to note is that SAT allows encrypted and authenticated messages to be exchanged directly between a network server and the SIM. Operators can use this functionality to configure the SIM, or to address specially developed third party applications residing on the SIM. The mechanism used to update data on the smart card is called "Data Download to SIM". The GSM 03.48 standard has enhanced this mechanism by providing security mechanisms to establish an end-to-end secured communication channel (mainly based on SMS). One end point of this secure channel is the SIM card and the other end is an Over The Air (OTA) server residing in the mobile operator's network.

IV. The Internet ID Approach

The emergence of identity management systems like Liberty Alliance, combined with the fact that around one billion GSM SIMs exist worldwide, has led to the development of the Internet ID approach. Internet ID tries to combine the best of both worlds: a standardized way to perform SSO according to LAP with the omnipresence of mobile phones and SIM cards. The approach proposes that the existing GSM authentication infrastructure (SIM plus corresponding network servers) should be used as a possible authentication method for LAP. The approach allows services on the Internet to be accessed via the mobile phone directly, but also enables the usage of the SIM as an authentication device for service access via a PC. The Internet ID approach differs from other SIM based authentication methods like EAP-SIM in a number of ways:

- It focuses on (but is not limited to) service level authentication at web sites rather than access network authentication.
- The SIM card can be used for authentication whilst residing in a normal GSM/GPRS/UMTS mobile terminal, even when services are accessed via a PC, i.e. there is no need to take the SIM out of the phone (or a second SIM provided by the operator) and put it into a separate smart card reader attached to or within the PC.
- No changes are required on the mobile terminal for any of the scenarios described. One scenario (Fixed Internet Service Access and Bluetooth) requires modifications on the SIM (SAT application) and the PC (software Plug-in installation), but these changes can be made remotely, so they are seen as less critical than a mobile terminal modification.

A. Basic Set-up

The elements of the Internet ID approach consist of a LAP SP and a LAP IDP, both of which are connected to the Internet. We assume that a mobile network operator operates the IDP, as only the mobile operator can authenticate the SIM card of the subscriber. The LAP Principal communicates with the SP either using a PC or a mobile terminal (which contains the SIM card). The Principal also exchanges information with the IDP (mainly for authentication purposes). The basic set-up is illustrated in Figure 1.

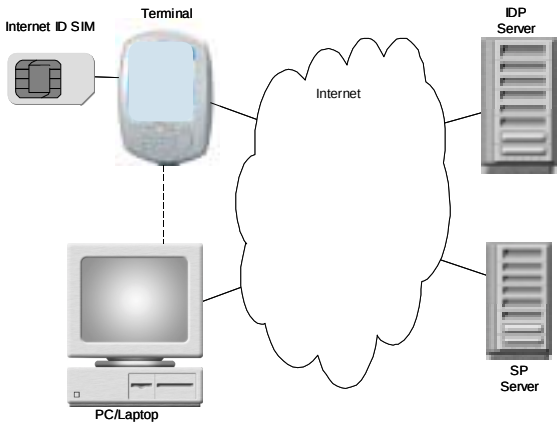


Fig. 1. Basic Internet ID set-up

Note that the mobile terminal is usually connected to the Internet via a mobile operator, not shown in Figure 1. Furthermore, it is possible to communicate locally between the PC and the mobile terminal via cable, Infrared or Bluetooth, as indicated with the dashed line.

B. Scenario: Mobile Internet Service Access

This scenario is relatively simple so it will only be described briefly.

A Principal tries to access a protected resource at an SP while browsing with the mobile phone (e.g. via WAP). If the Principal has not yet been authenticated, the SP re-directs his/her browser to the IDP. Since it is the Principal's mobile terminal that is re-directed to the operator's IDP server, it is simple for the operator to determine the Mobile Station Integrated Services Digital Network number (MSISDN) belonging to the terminal's SIM (e.g. through MSISDN forwarding). This MSISDN can then be used in a lookup-table to map it to a LAP username. This LAP username is then used to look up the previously agreed pseudonym to be used between the IDP and the SP for that particular Principal. The IDP generates an assertion for the successful authentication of the Principal (the authentication is implicit, since the user is already registered on the operator's network) and re-directs the Principal back to the SP. The assertion is either included in the re-direction message or a reference to it is sent, so that the SP is able to fetch it via a back end channel. After verifying the assertion, the SP grants the Principal access to the protected resource.

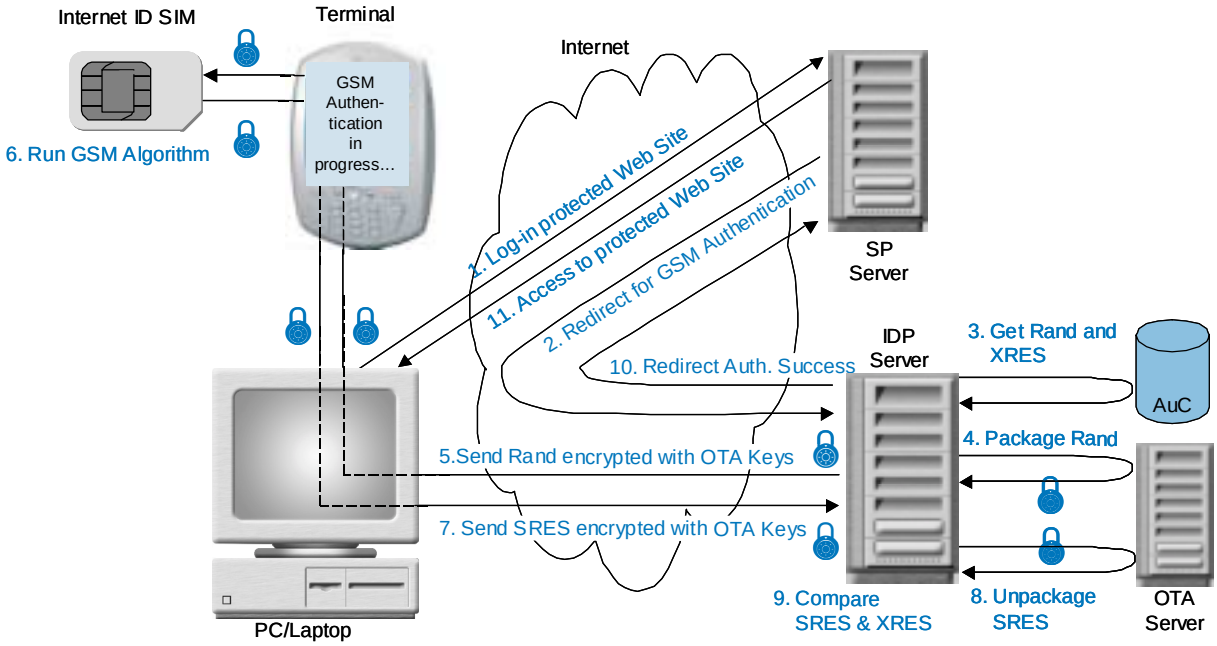


Fig. 2. Fixed Internet Service Access and Bluetooth – Message Flow

C. *Scenario: Fixed Internet Service Access and Bluetooth*

In this scenario a Principal tries to access a protected resource at an SP using a PC, laptop or other device without a SIM. The PC, laptop or other device does, however, have local connectivity to the Principal's mobile terminal (using Bluetooth in this example, although a cable or Infrared device could also be used). The set-up makes use of the fact that mobile operators typically have an OTA server in operation. The SIM can then be used for authentication in the following way (also described in Figure 2. The detailed message flow consists of the following steps:

1) The Principal tries to log in to a protected web site (this is typically the first step in a LAP message flow).

2) The SP re-directs the Internet browser to the IDP to check the user's authentication status. If appropriate, the authentication process is initiated (again a normal LAP step).

3) If authentication is required an "Internet ID" GSM authentication is performed. To do this, the IDP fetches the values Rand and XRES from the AuC.

4) The Rand is sent to the OTA server for encryption with the unique OTA keys of the Principal's SIM.

5) The encrypted Rand is sent via the Internet to the Principal's Internet browser and a browser plug-in forwards the encrypted RAND via Bluetooth to the SIM card in the mobile terminal.

6) The SIM decrypts the Rand using the locally stored OTA keys and invokes the Internet ID SAT application on the SIM. This application runs the GSM Algorithm (i.e. encrypts the Rand with the authentication key, K_i).

7) The result of this encryption (SRES) is encrypted with the OTA keys of the SIM and sent back to the IDP (via Bluetooth, browser plug-in, and Internet connection).

8) The IDP forwards the encrypted SRES to the OTA server to be decrypted.

9) The IDP compares the decrypted SRES to the expected response XRES.

10) If they match, authentication is successful and the message flow ends with the usual Liberty Alliance messages, i.e. the IDP re-directs the Principal's browser back to the SP (the re-direction containing an assertion or a reference to one).

11) The SP grants the Principal access to the protected web site.

Note that the authentication described in this scenario is not restricted to pure GSM authentication, as a SAT application in the SIM can provide more or less any form of authentication. Therefore, the method could easily be enhanced to use, for example, a digital signature based on PKI. Furthermore, the authentication method could include the use of a mandatory, application-layer PIN for cardholder verification.

D. *Scenario: Fixed Internet Service Access and Cellular*

If no local connectivity (via Bluetooth, Infrared or cable) is available, then the cellular network can be used to access the mobile terminal (and thus the SIM) for authentication. Due to the lack of local connectivity another form of "link" must be established between the Principal's mobile and the PC, in order to ensure that the Principal is really sitting in front of that particular PC. A typical method used to establish such a link between a mobile phone and a session on a PC is to send a random One-Time-Password (OTP) to the user's mobile terminal over the cellular network using SMS, for example. The user then has to transfer this OTP manually to the PC. Although this mechanism is easy to implement, it is not necessarily convenient for the end user because, in order to be secure, the OTP must be random and long, which makes the whole linking process both tedious and error-prone. Thus, a

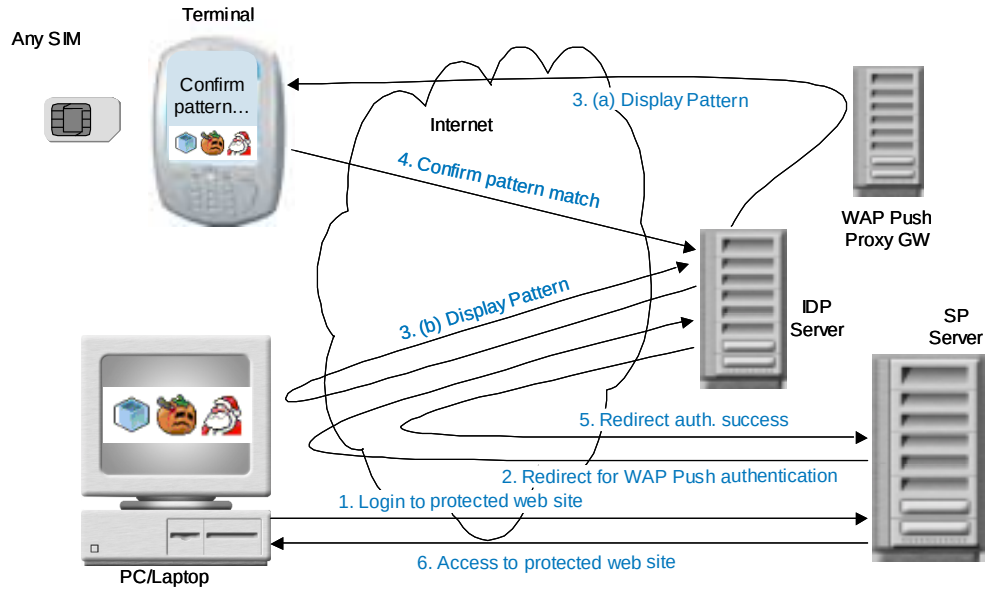


Fig. 3. Fixed Internet Service Access and Cellular– Message Flow

slightly different form of OTP was selected for the Internet ID approach. Instead of having to manually transfer the OTP from the mobile to the PC, the OTP is transferred to both devices in parallel. The only task for the user is to confirm that the OTP sent to the PC matches the OTP sent to the mobile terminal. To further simplify the matching process, OTP characters are replaced by symbols, as short sequences of symbols are easier for humans to match than longer sequences of characters. For example three symbols, chosen from a set of 100 possible symbols enables 1 million different OTP combinations (i.e. the same as a 6 digit number).

Figure 3 describes the message flow for the *Fixed Internet Service Access and Cellular* scenario in detail.

1) The Principal tries to log in to a protected web site.

2) The SP re-directs the Internet browser to the IDP to check the user's authentication status and, if necessary, start the authentication process.

3) If authentication is required a random pattern of symbols is generated, which is (a) pushed via a WAP Push Proxy GW to the correct user's mobile terminal and (b) simultaneously displayed on the PC/laptop.

4) The Principal compares the pattern received on the mobile device with the one displayed on the PC/laptop. If both match, the Principal confirms the match by, for example, clicking a respective link on the WAP page. To allow the IDP to distinguish between different confirmations, the invoked URL contains a unique identifier for each authentication request.

5) If the Principal confirms the match of the patterns, authentication is successful and the message flow ends with the usual Liberty Alliance messages, i.e. the IDP re-directs the Principal's browser back to the SP (the re-direction containing either an assertion or a reference to one).

6) The SP grants the Principal access to the protected web site.

Note that the described scenario makes indirect use of the SIM security, by relying on the correct delivery of the WAP Push message to the phone with the right SIM. Direct usage of authentication algorithms in the SIM (e.g. hash signing) is, in principle, also possible, provided the "linkage" between mobile and PC is ensured. Cardholder verification, which is also not included in the above scenario, could be provided through an application-layer PIN.

V. Conclusions and Future Research

Identity management systems like .NET Passport or the Liberty Alliance specifications provide users with convenient features like Single Sign-On and other identity-related services. A crucial task of these systems is to authenticate the participating users securely, yet conveniently. Today, authentication mechanisms are typically based on username and password combinations.

The Internet ID approach proposes the use of the SIM card (which is widely deployed) for authentication in Liberty Alliance based systems. This approach can be used when accessing services using either a mobile phone or a PC. For the case of the PC two different methods of accessing the SIM are described: a Bluetooth scenario, which provides the best user experience, and a cellular approach, which could be used in cases where a Bluetooth connection is not possible. The actual authentication mechanism on the SIM can be chosen flexibly. For instance, could it be based on GSM authentication or on alternative methods such as operator controlled PKI systems.

As a result, the Internet ID approach provides a secure and convenient authentication method for Liberty Alliance systems, re-using mobile phones and SIM cards as security tokens.

In the future the Internet ID approach could be extended in a number of ways:

- The SIM could be used for purposes other than authentication. For example, it could be used for other Liberty Alliance identity services such as attribute sharing (user consent via mobile phone, attributes stored in the SIM) or payments (payment authorization via mobile phone).
- The SIM could be used for authentication for services that are not accessed via a web browser. An example could be access network authentication (e.g. WLAN, DSL) or physical access to buildings.
- Approaches like EAP-SIM and EAP-AKA could be investigated with regards to their use for authentication for web site access.

In the long term such studies could lead to the mobile phone and the SIM becoming a universal security token for the user, independent of the service accessed or the device used to access the service.

ACKNOWLEDGMENT

The Internet ID approach is the result of collaboration between Ericsson, Giesecke & Devrient, and Vodafone. The authors would like to thank Monika Girnghuber, Silke Holtmanns, Jos Bosch, Najib Koraichi, and Alex Walter for their contributions to the Internet ID concept.

REFERENCES

[1] Microsoft .NET Passport web page, www.passport.com.
[2] Microsoft, ".NET Passport: Balanced Authentication Solutions", White Paper, April 2003.
[3] J. Hall Gailey, "What is .NET Passport", www.sys-con.com/webservices/articleprint.cfm?id=307.
[4] Liberty Alliance Project web page, www.projectliberty.org.
[5] Liberty Alliance Project, "Introduction to the Liberty Alliance Identity Architecture – Revision 1.0", March 2003.
[6] Liberty Alliance Project, "Liberty ID-FF Bindings and Profiles Specification, Version 1.2", April 2003.
[7] M. Mouly, M.-B. Pautet, "The GSM System for Mobile Communications", CELL & SYS, 1992.
[8] W. Rankl, W. Effing, "Smart Card Handbook", Wiley, 1997.
[9] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.
[10] H. Haverinen, J. Salowey, "Extensible Authentication Protocol Method for GSM Subscriber Identity Modules (EAP-SIM)", Internet Draft, April 2004.
[11] J. Arkko, H. Haverinen, "Extensible Authentication Protocol Method for UMTS Authentication and Key Agreement (EAP-AKA)", Internet Draft, June 2004.
[12] Wireless Identity Module Specification, WAP-260-WIM-20010712-a, http://www.wapforum.org/what/technical.htm.

Principles for Cellular Access Security

Geir M. Køien
Agder University College, Faculty of Engineering and Science,
Grooseveien 36, N-4876 Grimstad, Norway
Email: Geir.Koien@hia.no

***Abstract*— Cellular access security in 3G networks is based on an aging trust model. The integration of radio resource management (RRM), mobility management (MM) and the security protocols is suboptimal with respect to minimizing set-up delays. 3G cellular access security does not fully address the requirements and needs of the subscribers, the serving operator or the home operator. In this paper we present an analysis of 3G cellular access security and we propose a set of guiding principles for cellular access security.**

***Index Terms*–Cellular Access Security; Guidelines; Link Layer;**

I. Introduction

A. Emerging Requirements

Cellular access security was not a major concern for the first generation (1G) cellular systems. These system, like NMT, was originally designed without any security features. In NMT one initially had only a three digit password (k_1, k_2, k_3), which was sent in clear over the air. Due to problems with fraud one later added entity authentication during call set-up (NMT-SIS). The second generation (2G), which arrived during the early 1990s, was based on digital transmission. In GSM one had a simple *challenge-response* authentication mechanism and encrypted communication between the mobile phone and the base transceiver station. The security of the 2G systems has been a great success, but it now clear that many of the underlying assumptions of 2G security is not true anymore. In 3G security one has improved mutual entity authentication, confidentiality and integrity protection. But even 3G security has its shortcomings. Quite a few of the issues originates from the 3G system design. The problem is that the 3G systems inherits too many features and solutions from its 2G predecessors. At the same time, the requirements and environment that faces a 3G system is very different from the requirements and environment that much of the 3G inherited architecture base is designed to meet. These shortcoming also affect the access security. New and emerging requirements are not fully meet by the 3G cellular access security. Here we like to single out the delegated authentication inherited from 2G and the inadequate attempt at providing location privacy in 2G systems.

B. Paper layout

We start out we a brief presentation of 3G (UMTS) security in section II. In section III we examine some of the factors of the physical layer and the link layer that affects the design of cellular access security. In section IV we set out to define a set of principles for cellular access security. In section VII we provide a summary and some conclusions.

II. Access Security in 3G Systems

In this section we analyze access security in the 3G cellular systems. 3G security is an evolution from 2G security and this is a source of some shortcomings of 3G security. We use UMTS [1–3] as an example (access security for CDMA2000 [4] is relatively similar to UMTS).

A. The 3GPP Security Architecture

1) Entites and Principals: The main principals in the network is the Home Environment (HE), the Serving Network (SN) and the User Equipment (UE). The HE consists of at least one Home Location Register (HLR) and an Authentication Centre (AuC). The HE will keep all permanent data pertaining to the subscribers, including the the security credentials. The SN consists of core network servers (VLR/SGSN) and associated UMTS access networks (UTRAN). The UTRAN network consists of Radio Network Controllers (RNC) and Access Points (AP, also known as Node B). Cellular service provider networks commonly consists of both HE and SN components. From our perspective we shall not distinguish between "home SN" and "visited SN".

The UE is composed of a Mobile Station (MS), the Universal Subscriber Identity Module (USIM) and possibly a separate terminal entity. The USIM, which is an application running on the smartcard (UICC), contains the long-term security credentials for the subscription. The UICC/USIM is not normally a property of the subscriber, but is issued and controlled by the HE operator. Figure 1 gives an overview of the UMTS AKA.

2) Identification: The primary 2G/3G subscriber identity in the system is not the telephone number (ITU-T E.164), but the ITU-T E.212 International Mobile Subscriber Identity (IMSI) number. Identity presentation during over-the-air access precedes the AKA procedure that generates the session keys. The identity presentation is therefore transferred in cleartext on the over-the-air interface. This permits an adversary to track the subscribers. To mitigate the problem the VLR/SGSN will issue a temporary identity (TMSI). The UE will then present itself with IMSI only the first time it enters a service area (SGSN/VLR area). After encryption has commenced, the SN issues a TMSI number to the UE. The TMSI is only used by the UE in the non-correlated plaintext form. Use of TMSI thus provides a measure of location privacy. Unfortunately, the protection is shallow and is only effective against a passive adversary. An illicit network will easily be able to deceive the UE to present the IMSI. This is true even for the 3G case,

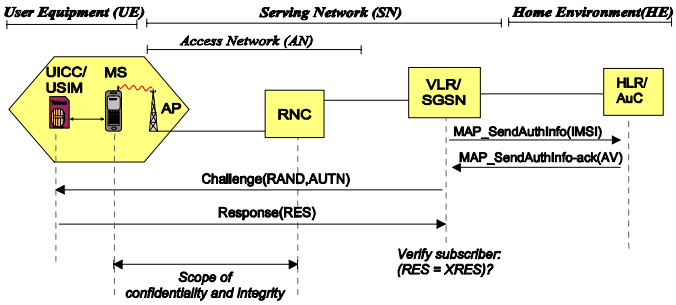


Fig. 1. UMTS Authentication and Key Agreement

because identity presentation (necessarily) takes place prior to authentication.

3) Authentication and Key Agreement (AKA): The AKA procedure can only be initiated by the network (SN) and is executed in two stages. First the security credentials (Authentication Vector - AV) is transferred from the HE to the SN. The second stage is a single-pass challenge-response procedure that provides mutual authentication through the use of a message authenticated challenge (based on ISO/IEC 9798-4 [5]). This procedure is executed locally between the UE and the SN. To avoid problems with replay attacks the AKA protocol requires a sequence no. management scheme. Confidentiality and integrity protection is provided between the MS and the RNC. The delegated AKA procedure is inherited from GSM, but for the 3GPP-WLAN interworking [6, 7] case a global AKA (HE-USIM) is used.

4) Link Layer Integrity and Confidentiality Services: Integrity protection is provided on all user related signaling messages between the MS and the RNC. The integrity function (f9 function) is under control of a 128 bit session key, IK. The cryptographic core is the KASUMI block cipher (w/64 bit block length). The produced ICV is 32 bits long. Confidentiality protection (f8 function) is by means of the KASUMI block cipher in a modified OFB mode and under control of a 128 bit session key, CK. The f8 function is applied to all user related data (including system signaling).

5) Network Domain Security: With a two staged AKA procedure it is imperative that the transport of security credentials from HE to SN takes place in a secure manner. The SS7-based MAP protocol [8] is the AKA transport protocol for the HE-SN stage. To protect MAP one must either use the MAP security extensions (MAPsec, [9]) or have MAP transported over IP (which is now possible). For IP one have developed an IPsec profile for use in 3GPP called NDS/IP [10].

B. Shortcomings of the 3GPP Security Architecture

1) Legacy Problems: UMTS has inherited may features and restrictions from its ancestor GSM. The GSM system was designed to be a mobile equivalent of ISDN, and there is much focus on providing CS communication. In the radio access one has the DTAP (inspired by the ISDN DSS.1 protocol) and LAPD protocols, and in the core network one has ISUP and the SS7 based MAP protocol. These protocols were excellent

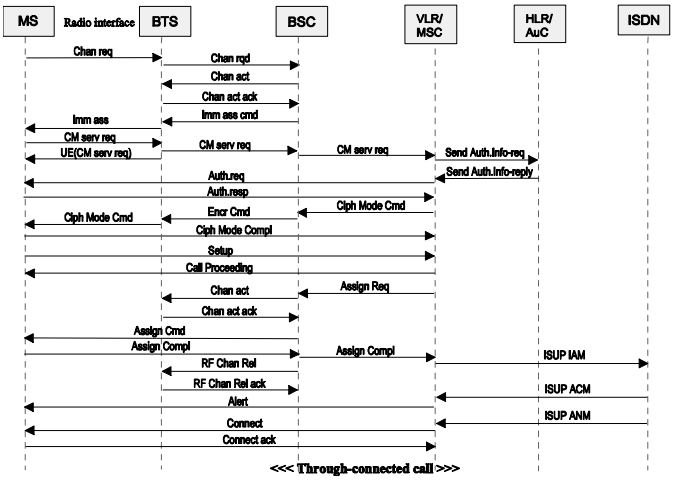


Fig. 2. GSM Call set-up (simplified)

choices for GSM, but are not appropriate choices for networks which aims to deliver PS services. An example is the call set-up model in GSM (fig.2). Due to limited channel capacity and size restrictions on signaling messages in the GSM radio access, multiple round-trips over the radio interface is required to establish a user-data traffic channel. To some extent this cannot be avoided, but in GSM the problem is exacerbated by an inflexible channel arrangement and lack of functional integrations of the RRM, MM and security procedures. The problems with the channel arrangement is an unfair criticism of GSM, which has a narrow channel (200 kHz) TDMA carrier. For UMTS, which has a wideband (5 MHz) radio system, the criticism is justified. The sequential execution of the RRM, MM and security procedures, which to some extent was a practical necessity in the narrowband GSM system, is performance-wise an inappropriate solution for the wideband UMTS radio system.

The problem is also present in the Core Network (CN). The MAP payload is restricted by the SS7 link layer limit (272 octets). The result is multiple round-trips (and link layer segmentation overhead) were a single round-trip would have sufficed given a more capable transport network. This makes communication setup expensive and time consuming, which is particularly bad for PS services.

2) An Aging Security Model: The UMTS security model dates back to the design of GSM in the mid eighties. Back then in Europe, there were only national operators and the trust model was not a big issue. Trust can be viewed as the intersection of beliefs and dependence. During roaming the HE and the UE will necessarily have to trust the SN to some extent. One depends on the services provided by the SN, and one hopefully have justified belief in the compliant behavior of the SN. With the 2G/3G delegated authentication the HE is in a situation were the balance between beliefs and dependence is uneven. The GSM Association reports that it has (July 2004) more then 620 operator members from 205 nations. In this environment the notion of delegated authentication

seems naive and outdated. Mechanisms to strengthen the home control seems a legitimate demand.

In the case of 3GPP-WLAN interworking one have addressed this issue, and for the 3GPP-WLAN context there is global authentication [6]. But this leaves the SN almost without authority. One must take into account that both the UE, SN and HE are legitimate parties in the security context.

Principle-1: All principals must be fully included in the security contexts.

3) Key Distribution: We have two levels of key distribution in the 3G system. At the top level we have the security credential (AV) transport from the HE to the SN. The transport protocol is the aging MAP protocol. It is possible to secure the MAP protocol with the MAPsec extensions [9], but MAPsec has some problems of its own. The most problematic issue with MAPsec is nevertheless the complications with configuration and management. It is very difficult to phase in use of MAPsec in the legacy MAP environment. The MAP protocol may also be transported over IP, and in this case one may use IP security mechanisms [10, 11]. The problem with backwards compatibility will still be present for communication with SS7 based MAP networks.

The second level of key distribution concerns transfer of session keys to the terminating points. Confidentiality and integrity is provided between the MS and the RNC. We observe that since the key agreement points (USIM and VLR/SGSN) differs from the key usage points (MS and RNC). One must therfore have secure transport from the USIM to the MS and from the VLR/SGSN to the RNC. This is a weak point of the 3G architecture, and the problems are not properly addressed. There is a non-mandatory mechanism (NDS/IP, [10]) available for the Iu-interface (VLR/SGSN - RNC), but this solution is only available for networks with an IP based Iu-interface (most legacy networks still have the SS7 base intact). The USIM - MS interface has traditionally been seen as an internal interface. But, the trend now is for a functional split in which the UICC/USIM may not be physically present at the MS (there may be multiple radio units: UMTS, WLAN etc). The interface may then be implemented over a Bluetooth link or similar. Secure session-key transport from the USIM to to MS is then an issue.

Principle-2: Key generation should preferably be at the key usage points. Keys must never be transported in cleartext.

The second part of principle 2 should be self evident, but we feel it must be explicitly stated since the current 3G architectures still transfers keys in cleartext.

4) Location Privacy: Privacy a growing concern. The use of TMSI in UMTS provides some privacy protection, but the protection is shallow. A false network can easily circumvent this protection. Since the SN can trigger clear-text transfer of IMSI without authenticating itself, the protection is only effective against passive adversaries. We must therefore conclude that UMTS access security does not offer effective identification and location privacy.

The problem is rooted in the fact that the IMSI identity is not only an identity, but also an address. For the initial registration at a new SGSN/VLR, the network needs to have the IMSI number. It uses the IMSI number to compose a ITU-T E.214 Mobile Global Title (MGT) address. The MGT is then used by the VLR/SGSN as an address for the HLR, and only by using this address the VLR/SGSN can get the subscriber data and security credentials from the HLR.

5) Coverage and Strength of Protection: UMTS does not offer integrity protection of user data and in retrospect this must be considered an omission. The choice of a 32 bit ICV is a weakness, but one must keep in mind that the lifetime of the signaling messages is short and that falsifying a messages also includes producing a valid message in the signaling context. The choice of a cryptographic core with only 64 bit block length is also a weakness. A block cipher key should not normally be used to encrypt more than $2^{b/2}$ blocks, where b is the block size. For KASUMI, this would impose a limit of approx. 32 Gigabytes of data to be encrypted with any one key. In UMTS, the HSDPA service provides (downlink) bit rates exceeding 10 Mbps, which means that this is not purely a theoretical shortcoming (At 10Mbps it will take less than 8 hours to exhaust the 32 Gigabyte limit). We note that UMTS is designed to allow for new f8/f9 functions, but backwards compatibility concerns will arise and the KASUMI based f8/f9 functions will stay in the networks for years to come.

III. Radio Resource Management, Mobility Management and the Security Protocols

Successful security mechanisms must be both effective and efficient. We focus on efficiency, and interpret it primarily as requirements on low latency in the security context set-up.

The 3G radio systems are not really designed for short-lived connectionless communication. This is visible in the choice of CDMA and WCDMA as the basis for the radiosystems, but it is even more pronounced for the radio resource management scheduling. The RRM scheduling is to a large degree optimized for connection-oriented circuit-switched communication. While there are many differences between GSM and UMTS, the call set-up depicted in fig.2 is to a large extent identical to the call set-up in UMTS. A main reason for this is that the signalling machinery in the GSM DTAP protocol is kept in UMTS [12].The focus on circuit-switched communication also affects the mobility management procedures, but not to the same extent as for the RRM scheduling. The main problems here is the fact that the procedures are overly sequential. This was a necessity in GSM, but the GSM signaling state machine was kept and the potential for integration of signalling procedures was not realized.

A. The Physical Layer

We now investigate some properties of the physical layer that (may) affect the security procedures.

1) Modulation: Modern radio systems are flexible and modulation schemes can often be chosen per carrier/subcarrier. For signaling a relatively conservative modulation scheme is needed.

2) Coding and Error detection/correction: For system signaling it is imperative that there are no residual errors in the control plane messages. One tend not to include the strongest error correction for signaling. Instead one relies on a robust error detection mechanism to avoid accepting (radio) corrupted messages. Low latency is first priority and retransmission is therefore not applicable (retransmission itself is a link layer feature). Coding schemes for signaling are usually a combination of convolutional coding and block coding. Turbo codes, which is a hot topic in coding theory, is generally not suited for applications that cannot tolerate long propagation delays.

3) Interleaving: Interleaving induces a delay in the processing as multiple frames must be present before de-interleaving can take place. For transaction-oriented services the penalty can be substantial (for sustained throughput applications w/large window size the effect is negligible). Interleaving depth is a design decision, and an example from GSM illustrates this. The interleaving used for data on a full-rate traffic (TCH/F) channel induces a delay of more than 100 ms, while the shallow the interleaving on the dedicated signaling channel (SDCCH) only induces a 14 ms delay.

4) Bandwidth: We need "sufficient" bandwidth. This requirement is related to the need for sufficiently large MTU size. Performance-wise it is undesirable to have MTU segmentation for signaling messages. During the set-up phase segmentation is commonly not allowed.

5) Physical channel arrangement: What we want is a wideband channel arrangement with flexible modulation and coding. Furthermore, we want availability of both narrow low-capacity channels and large (multiplexed) channels. Finally, we want fine grained temporal control on the allocations. To support this one will need a flexible scheme like OFDM/OFDMA (fig.3)

The channel arrangement in OFDMA systems is based on wideband carriers that are divided into sub-carriers. These sub-carriers are again divided into sub-channels (time/frequency). This scheme is flexible and allows allocation of both wideband data channels as well as narrow channels. To complement the picture the various channels/sub-carriers can have different power levels, error protection (incl.interleaving depths) etc. The possibility of making fine-grained allocations makes OFDMA an excellent choice for provision of PS services. The high resolution in both time and frequency allows for CS-like channel reservations and short burst-like high capacity allocations. The newer Wi-Fi standards (for instance IEEE 802.11g) and the upcoming WiMAX (IEEE 802.16a) standard all include OFDM technology.

B. Link Layer - Radio Resource Management

1) Retransmission: Retransmission at the link layer is for services that cannot tolerate biterrors or were the cost of handling errors at higher layers (e.g. TCP) is too high. Retransmission can induce significant delays and will add payload overhead. Retransmission is therefore not used for lower layer control channels.

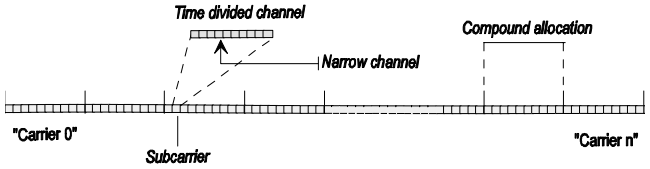


Fig. 3. OFDMA channel structure

2) Channel granularity: The present FDMA, TDMA and CDMA schemes are inflexible with respect to channel allocation.

On the other hand, OFDMA systems do provide the necessary flexibility. One can then allow UE-SN to have a permanent low capacity (low latency) control channel. This is attractive for PS environments, and it will permit faster set-up of new sessions and improved management of security contexts. A permanent control channel is also useful for fast channel allocations to facilitate high quality PS services. This will impact the security model in that immediate availability of a security context is crucial for the perceived service quality. Thus, the security context must be established prior to demand or it must be generated on the fly.

3) Link Layer Address Schemes: Link layer address information will be exposed in the RRM procedures. This cannot be avoided, but unless care is taken it will allow an adversary to track the subscribers.

4) RRM Integration: There is potential for integration of RRM and security procedures, and in particular for session establishment and keying/rekeying procedures. There is also room for improved integration of handover and security context transfer procedures. Integration of selected RRM and security procedures will reduce the no. of round-trips.

C. Link Layer - Mobility Management

Many control issues and service model requirements are intrinsic in the design of the MM mechanism. The control model is essential to the security architecture.

1) Control Aspects: The traditional relationship between the principals (UE,SN,HE) is asymmetrical. The HE has jurisdiction over the UE, and the HE and SN has mutual (roaming) agreements. By assuming transitive trust, the UE and SN can interact. The relationship between SN-UE is inherently unbalanced. The SN unilaterally controls channel allocations and relocation (handover). On the other hand, the UE is in principle entitled to choose which SN to attach to, though it is restricted by HE-SN roaming agreements.

One may have control models in which the HE functionality is under jurisdiction of the user (Mobile IP is an example, [13]) and were control decisions (session transfer etc) are made by the mobile node. Whatever the control model may be, it is crucial that the security architecture reflects the trust model

that is assumed by the control model. In this paper we assume a traditional cellular system control model.

2) Micro- and Macro Mobility: We define micro-mobility to be the procedures necessary to allow seamless radio channel transfers. Mobility is managed by coordination of resource allocation at the physical and link layer to provide transparent handovers between radio channels. Micro-mobility, by its very nature, requires close coordination or the radio resources. One often confine this type of mobility handling to homogenous access network architectures, but we observe that micro-mobility management can also be had for heterogenous access networks provided one can synchronize and control the resources closely.

Macro-mobility is normally defined as the handling of mobility issues at the network layer. This makes close control of the radio environment almost impossible, but it is often the only means for handling heterogeneous access technologies. Macro-mobility management is global in scope with respect to addressing schemes etc and is therefore well suited for handling connection reestablishment and non-time critical session transfers.

3) Reestablishmen/handover: Transparent handovers requires infrastructure resource coordination, and it is probably best that the access network be in control of transparent link layer handovers. Session reestablishment is easier to provide, but it is inappropriate for streaming services (like for instance speech). We assume infrastructure supported seamless handovers.

4) MM Integration: The potential for integration of MM and security procedures is good, both for the SN-UE procedures and the HE-UE procedures. We note that in particular the registrations (location updating) and the security context establishment (and renewal) are good candidates for integration. Even loose integration with logically decoupled messages contained in the same packet will achieve substantial improvement in latency as redundant round-trips are avoided.

Principle-3: In order to reduce the required no.of signaling round-trips, selected RRM, MM and security protocol procedures should be integrated.

IV. Design of Cellular Access Security

A. The Need for a Hierarchical Structure

One may classify the security relations according to either spatial or temporal coverage, but in practice the two views will yield similar structures. We have three basic levels:

- **Permanent–Global.** There are two types of Permanent–Global security contexts.
  A) The roaming agrement between the HE and SN. This security context covers the necessary security credentials to set up secure communications channels between the HE and SN entities. The HE-SN roaming agreement normally applies to all HE subscribers and for the full SN area. The lifetime of the agreement is normally in the order of years.
  B) The subscription context between HE and UE. The lifetime of the HE-UE context is permanent with respect to the subscription contract. The coverage is normally global in the sense that the UE is permitted to roam all the networks that the HE has roaming agreement with.

- **Medium–Term.** Here we potentially have three contexts.
  A) The HE-UE context. This medium-term context covers authentication between HE-UE and the production of associated medium-term security credentials. The security credentials is to be the basis for production of session keys for communication directly between the HE and the UE. We note that lawful interception requirements may demand that the SN be able to incept the communication. The HE-UE context may be used to secure HE configuration of UE data etc, but there may also be requirements for tunneling all user related communication back to the home network.
  The validity of this context would typically be locked to the UE presence in the *service area* (associated with a server (VLR/SGSN)) or with a *location area* (associated with a radio controller). The validity should also be restricted by a maximum time and by a maximum usage (represented as the maximum derived session key sets).
  B) The HE-SN context. This context would use the security credentials from the roaming agreement context. One may then have mutual entity authentication between nodes in the HE and SN networks. The respective nodes will then have 1-to-1 medium-term context. Since this context is independent of the UEs, we expect the contexts to be managed independently of any particular roaming/security event.
  Note: We shall consider further analysis of the HE-SN context to be outside the scope of this paper since it is independent of UE related events.
  Additionally, one *may* decide that it is beneficial to have a dedicated per-UE protected channel between the HE and the SN. The validity of the dedicated context would similar as to the HE-UE medium-term context.
  C) The SN-UE context. This context covers authentication between SN and UE. It includes medium-term security credentials to be used as basis for the short-term context. The validity of the SN-UE context would be similar to the HE-UE context.

- **Short-Term**. The short-term contexts are concerned with session keys. We have three contexts.
  A) The HE-UE context. This context contains the session keys and is derived from the medium-term credentials. The validity of the HE-UE short-term context is limited to the lifetime of the associated medium-term context, but may be significantly shorter.
  B) The HE-SN context. See note above.
  C) The SN-UE context. This context contains the session keys and is derived from the medium-term credentials. The validity of the SN-UE short-term context

is limited to the lifetime of the associated medium-term context, but may be significantly shorter. We expect rekeying (context renewal) to occur when one change access points, for periodic location updating and other RRM events (like bandwidth changes, QoS changes etc).

Principle-4: A hierarchical security context structure is needed for cellular access security.

1) Separation of Access Networks: Modern cellular systems is expected to support multiple access methods. The SN must therefore be prepared to support multiple heterogenous access networks with potentially different levels of physical security of the network elements, differing coverage of protection etc. The actual security level of the different access networks will vary correspondingly, and it is therefore unsound practice to use the same security credentials for different access networks.

Principle-5: The UE-SN must maintain separate medium-term security contexts for separate access networks.

B. Computational Balance

1) Triggering Events: Triggering of access security events cannot be controlled by the principals. They must therefore be able to execute the security protocols on demand.

2) Secure Storage Card: The secure storage cards (smartcards) have limited processing power and limited storage space, but they frequently include HW support for the crypto-primitives. The basic requirement is to compute the AKA functions on-the-fly. Modern smartcards can do this with relative ease.

3) The Mobile Station: The MS must be able to run the confidentiality and integrity algorithms without delay when the MS sends/receives data at maximum speed. In practice, the processing requirements for the security algorithms are modest. The primitives are typically implemented in HW to increase processing speed and to reduce power consumption.

4) The Serving Network: The 3GPP AKA does not require the SN nodes to do much during the AKA since the key material and response value is pre-computed by the HLR/AuC. For improved AKA protocols the SN should be an active part in the AKA processing. However, the SN nodes will serve a large number of users and must be able to execute the AKA protocol on-demand. The instantaneous processing requirements may therefore potentially be quite demanding. It may be beneficial if the SN is allowed to delay non-time critical sequences. Session security context renewal is a sequence were the parties have some freedom in delaying the sequence as long as it can be assured to complete before the current context expires.

When it comes to confidentiality/integrity related processing we have much the same case as for the MS. We assume that the crypto-processing will relatively be modest (compared to the radio-related processing).

5) The Home Subscriber Server (HSS): A 3GPP HLR/AuC may serve a huge no. of subscribers. To instantly compute session credentials for all of them will create a substantial load. For 3GPP AKA the HLR/AuC can pre-compute AVs for the subscribers. Peak processing performance is therefore not too important as long as the average capacity is sufficient. For mobile networks with distinct *busy hour* conditions this is a considerable advantage.

For a post-3G HSS the situation is more difficult. AKA events are time critical and the HSS may not be afforded the luxury of pre-computation. The HSS must then be dimensioned for a higher instantaneous crypto-processing load than the 3GPP HLR/AuCs. Still, with optimized crypto-primitives implemented in HW we postulate that the capacity required need not be excessive. Due to Moore's law we expect the relative AKA processing requirements pr subscriber to be lower for a post-3G HSS than it was for the AuC during GSM rollout in the early 1990s.

C. Communication Capacity Balance

1) Radio Interface (MS-AN): For a shared radio interface one will necessarily have capacity restrictions. These restrictions will not affect the modest capacity requirements for signaling, but there are commonly restrictions on signaling message size during the initial phase of set-up events. This may preclude support for primitives that requires large information elements (IEs) (e.g. Diffie-Hellman based key generation). Furthermore, support for data expanding ($E_K(M) \rightarrow C$, *where* $|C| > |M|$) primitives is problematic.

2) Other Interfaces: The remaining interfaces will be fixed network interfaces with permanent resource allocations on the link layer. The message size restrictions will then be determined by the MTU size of the link layer and/or message size restrictions on the network layer. In practice, these restrictions pose no problems for our signaling needs (the common minimum is for a MTU size of 1500 bytes).

Use of a conventional Diffie-Hellman exchange for production of shared secrets of 256 bits will require exchange of data elements with a length of approx. 15000 bits, but even this is permissible for fixed line connections.

D. Cryptographic Primitives and Key Strength

We assume that the primitives are effective and that they are used appropriately. It has been argued that one today should design systems for 128 bit security [14]. This is fine for confidentiality since we already have good block cipher primitives with 128 bit blocksize controlled by a 128 bit key. The goal is less practical for integrity protection. To have 128 bit worth of "collision avoidance" one need, due to the birthday paradox, as a minium a 256 bit key. The produced integrity check value (ICV) should also be 256 bits. However, for link layer security this is neither necessary nor feasible. The messages are short and its totally impractical to add a 256 bit ICV to each message. Considering the validity time of link layer messages it is also completely overkill.

Principle-6: Confidentiality and Integrity services should be pervasive and available to both control plane and user plane transfers.

1) Confidentiality: We only consider symmetric primitives. Block cipher primitives in stream mode (output-feedback or counter mode) will likely be preferred, but we do not exclude stream ciphers.

Principle-7: Confidentiality; Key size for confidentiality algorithms should be 128 bit. Block size (inner state) should be 128 bit.

2) Message Authentication / Integrity: We only consider symmetric primitives. We do not have a strong preferences on whether the MAC function should be constructed from ciphers or from cryptographic hash functions.

Principle-8: Integrity; Key size for message authentication should be 128 bit. Integrity check value size should be at least 64 bit.

The above recommendation is for *link layer* integrity protection. The frame/message size on the link layer is short and adding more than 64 bits probably cannot be justified. Also, the validity time of link layer frames is very short and thus there is no need for protection that can withstand attacks with years of compuation time.

E. The Basis for AKA

The UMTS AKA is based on a long-term 128 bit shared secret, K, stored at the USIM and the AuC. For the accumulated number of invocations of the AKA algorithms during the lifetime of the UICC/USIM, the length of K seems sufficient. The average lifetime for GSM SIM is modest (1-3 years). Even if we assume a lifetime of 5 years and 100 AKA invocations pr day, the total no. of AKA invocations will be less than 90000 during the lifetime of the UICC/USIM. This will not exhaust K. We note that the USIM will only respond to valid challenges. One cannot "drive" the functions as one can for GSM AKA.

UMTS uses MAC functions for the AKA functions, and use of one-way functions to derive key material seems more attractive than to transfer keys to the UE. The alternative to pre-shared secrets is use of public-key (PK) key pairs. We note that ECC, with relatively short keys and fast execution, is an excellent choice for wireless security [15]. To use a combination of PK signatures, PK encryption and hash functions is attractive since it provides more flexibility than the use of pre-shared secrets does.

F. Location Privacy

To provide location privacy functionality one must hide the correlation between the logical subscriber identity and the presented link layer address/identity. The mechanism currently used by 3G networks is insufficient and can be circumvented by an active attacker. The root of the problem is that the permanent identity, IMSI, is also an address. During initial registration the IMSI must be in clear to the SN since the SN needs to derive the HLR address from the IMSI. Without this address (the Mobile Global Title) the SN cannot request the subscriber data or the security credentials from the HLR/AuC.

Principle-9: UE identity and UE address information should be decoupled.

Principle-10: The UE identity should never be exposed on the radio interface.

Decoupling alone does not solve the problem, and the UE identity must be made available to the HE. However, there is no compelling reason for the SN to know the UE identity provided it has assurance from the HE that it acknowledges the UE. The justification here is that the HE has full jurisdiction over the UE with respect to the security credentials and to charging/billing. The SN does not need to know the true identity of the UE provided that the HE accepts responsibility for the UE.

Principle-11: The permanent UE identity should not be revealed to the SN.

Registration in a new service area is always triggered by the MS/UE. This event will also trigger the need for establishment of a security context between the SN and the UE. In order to manage the security context and the channel allocations one needs a reference. This reference (*TREF*) only needs to be a temporary link layer reference for communication between the UE/MS and the SN/AN. It is possible for both the UE and the SN/AN to generate the *TREF*. We note that if the UE is to generate *TREF*, then the *TREF* information element will have to be generated at random and it must have sufficient size for collisions to occur at a very low probability. Given that a service area potentially can encompass millions of subscribers, we must require the bit length of the *TREF* to be comparatively long. However, even if the service area contains a billion active users, a 64 bit *TREF* will make the probability so small as to be completely insignificant. We therefore postulate that the temporary identity (*TREF*) can be selected by the UE.

To maintain location privacy, we must additionally require that the *TREF*, combined with radio context information, does not permit the an adversary to derive the true user identity. As has been mentioned, the *MM Registration* procedure is triggered by the UE. If this procedure is integrated with the *medium-term* security context set-up, one can optimize the signaling and still provide full location privacy. This would require the UE to be the initiator of the AKA protocol and the entity that chooses the *TREF*.

Principle-12: The UE shall be the initiator of the AKA protocol for medium-term security context establishment.

Another aspect of location privacy is whether or not the HE needs to know the location of the UE. We postulate that the HE does not have a legitimate need to know the precise location of UE for execution of AKA. See [16] for a proposal on how to limit HE knowledge of UE position. We must balance this against the legitimate need the HE has for "home control" [17]. The SN will always be able to deduce UE location based on radio information, but this is permissible since we have stated that the SN shall not know the permanent UE identity.

Principle-13: The HE shall not be given precise UE location information.

We note that the *service area*, which is associated with the SN server (VLR/SGSN), will likely be known to the HE. The *location area* would not normally be known to the HE, but unless this area is very small it may also be permissible to let the HE know the location area code.

V. Formal Verification

Correctness, both at the (syntactical) protocol level and (semantic) security logic level, is an absolute condition. Model checking tools are very appropriate for communication protocols. Tools like Promela/SPIN [19] are excellent for verifying state-machine properties, but have weak support for verification of security properties.

Security properties have traditionally been verified by logic based tools and techniques. The IST AVISPA project have successfully managed to develop model checking tools for security verification. An example here is the OFMC [20] tool which is both efficient and effective at verifying security protocols.

However, no single tool/technique will suite all needs. We therefore advocate the use of multiple formal verification methods since the various techniques and tools have different strengths and weaknesses.

Principle-14: Formal verification of security protocols is essential for our trust in the protocols.

VI. Summary

1) Flawed 3G Security: Our analysis shows that the current 3G cellular access security architectures are flawed. The problems are related to system features and restrictions inherited from the 2G antecedents, including an outdated 2G trust model, a 2G system architecture and service model focused almost exclusively on CS service provision and finally technical restrictions due to the now obsolete SS7 signaling protocols.

2) Improved Integration: Future cellular system will have better and more flexible radio networks. This will allow for designs with integrated RRM, MM and security protocols, and one may then be able to provide near instantaneous access. To be able to realize this potential one needs to rethink the cellular access security model and to develop new security architectures in conjunction with the RRM and MM functionality.

3) Design Principles: The set of identified principles and remarks collectively form a a high-level set of guidelines and observations for cellular access security. This is in the spirit of the principles in [18], although here applied to a specific domain. The principles constitute a valuable input for the design of a cellular access security architecture, but they are no substitute for formal verification.

4) Conclusion: We have analyzed the current 3G access security and found it wanting. From a performance perspective, better integration with RRM and MM procedures is necessary. The full potential will only be realized for high-capacity radio networks with flexible channel management.

We believe that our stated principles can form the basis for design of modern cellular access security architectures. However, we do not claim that our set of principles are exhaustive or that following them will necessarily lead to an efficient and effective security architecture.

References

[1] 3G TS 33.102: *3G Security; Security architetcure (Release 6)*, 3GPP, Sophia Antipolis, France, 2004

[2] G.M. Køien, *An Introduction to Access Security in UMTS*. IEEE Wireless Communications Mag., Vol.11, No.1, pp.8-18, Feb. 2004

[3] K. Nyberg and V. Niemi, *UMTS Security*. ISBN 0-470-84794-8, Wiley, 2003

[4] G. Rose and G.M, Køien, *Access Security in CDMA2000, Including a Comparison with UMTS Access Security*. IEEE Wireless Communications Mag., Vol.11, No.1, pp.19-25, Feb. 2004

[5] ISO/IEC 9798-4: *Information technology - Security techniques - Entity authentication - Part 4: Mechanisms using a cryptographic check function*. ISO, Geneva, Switzerland, 1999

[6] 3G TS 33.234: *3G Security; Wireless Local Area Network (WLAN) Interworking Security (Release 6)*. 3GPP, Sophia Antipolis, France, 2004

[7] G.M. Køien and T. Haslestad, *Security Aspects of 3G-WLAN Interworking*. IEEE Communications Mag., Vol.41, No.11, pp.82-88, Nov. 2003

[8] 3G TS 29.002: *3rd Generation Partnership Project; Technical Specification Group Core Network; Mobile Application Part (MAP) specification;*, 3GPP, Sophia Antipolis, France, 2004

[9] 3G TS 33.200: *3G Security; Network Domain Security; MAP application layer security (Release 5)*. Sophia Antipolis, France, 2002

[10] 3G TS 33.210: *3G Security; Network Domain Security; IP network layer security (Release 6)*. Sophia Antipolis, France, 2004

[11] S. Kent, and R. Atkinson, *Security Architecture for the Internet Protocol*. IETF RFC 2401, Nov. 1998

[12] 3G TS 24.008: *3rd Generation Partnership Project; Technical Specification Group Core Network; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3*, Sophia Antipolis, France, 2004

[13] D. Johnson, C. Perkins and J. Arkko, *Mobility Support in IPv6*, IETF RFC 3775, June 2004

[14] N. Ferguson and B. Schneier, *Practical Cryptography*, ISBN 0-471-22357-3, Wiley, 2003

[15] K. Lauter, *The Advantages of Elliptic Curve Cryptography for Wireless Security.* IEEE Wireless Communications Magazine, Vol.11, No.1, pp.62-67, Feb. 2004

[16] G.M.Køien and V.A.Oleshchuk, *Privacy-Preserving Spatially Aware Authentication Protocols: Analysis and Solutions*, Proceedings of NORDSEC 2003, pp.161-173, ISBN 82-993980–4-5, Gjøvik, Norway, 2003

[17] G.M.Køien and V.A.Oleshchuk, *Spatio-Temporal Exposure Control; An investigation of spatial home control and location privacy issues*, Proceedings of IEEE PIMRC 2003, pp.2760-2764, Beijing, China, 2003

[18] M. Abadi and R. Needham, *Prudent Engineering Practice for Cryptographic Protocols*. DEC SRC Research Report 125, California, USA, June 1994

[19] G.J. Holzmann, *The Spin Model Checker : Primer and Reference Manual*. ISBN 0-321-22862-6, Addison-Wesley, Sept. 2003

[20] D. Basin, S. Mödersheim and L. Viganò, *An On-the-Fly Model-Checker for Security Protocol Analysis*. In *Proc. of ESORICS 2003*, LNCS 2808, Springer-Verlag, Oct. 2003

On Private Similarity Search Protocols

Sven Laur and Helger Lipmaa
Laboratory for Theoretical CS, Department of CS&E
Helsinki University of Technology, P.O.Box 5400, FIN-02015 HUT, Espoo, Finland
Email: {slaur,helger}@tcs.hut.fi

***Abstract*— In a private similarity search (PSS) protocol, a client receives from the database the entry, closest to her query, without either the client or the database getting to know more information than necessary. PSS protocols have potentially wide application in areas like bioinformatics, where precise queries might be impossible. We show that the previously proposed PSS protocols by Du and Atallah have serious weaknesses; in particular, some of their protocols can be broken by a semi-honest third party who observes a relatively small amount of traffic. In several cases, we show that even maximally securified versions of these protocols—when used as proposed by Du and Atallah—are not private in the sense, needed in the practice. We propose a few protocols that are better from the privacy viewpoint, but none of the proposed protocols is really efficient.**

***Index Terms*—Cryptanalysis, cryptographic protocols, privacy-preserving data-mining, private similarity search.**

I. Introduction

In a private similarity search (PSS) protocol [1], a client receives from the database (the index of the) the entry, closest to her query, without either the client or the database getting to know more information. Similarity search is used in many cases where, e.g., finding the exact match is impossible or infeasible, when the data is corrupted by noise or the user is really interested in similar objects. A canonical application area of the PSS is bioinformatics, together with related fields like biometrics. A motivating task could be, given a DNA sample of a criminal, to find the closest match in the genome database without compromising the safety of honest citizens. To implement such applications, one must address the privacy concerns. Otherwise, both clients and database maintainers would be discouraged to participate in such services.

One can expect private similarity search to be a hard problem, in particular since at some point during the PSS protocol, the participants have to find minimum over non-public distances. Private minimum finding is a well-known cryptographic hard problem and only generic inefficient protocols for it are known [2].

Thus, one cannot expect to design really practical protocols for the PSS that are secure in the sense of secure two-party computations [3]. The main goal of published research on the PSS [1] has been to propose *efficient* PSS protocols that are secure according to somewhat less stringent notions. For example, in [1] the authors propose several PSS protocols that are relatively efficient but make use of a conditionally trusted third party Ursula, who is assumed to follow the protocol and not to collaborate with any other party. Moreover, the protocols from [1] are only claimed to be secure against *ciphertext-only attack.*

Our contributions. We show that several protocols in [1] are insecure even following the weak security definitions of their authors. First, [1] proposes protocols for the PSS with respect to the Euclidean distance. This protocol essentially employs a private minimal scalar product (PMSP) protocol, using the help of Ursula to find the minimum. The PMSP protocol masks the real distances by using *additive masking functions*.

We show that the PMSP protocol of Du and Atallah [1] is completely insecure against ciphertext-only attacks. Namely, we show that a conditionally trusted third party Ursula can recover the queries by observing a small amount of traffic and using a straightforward matrix equations. Our attack against this protocol succeeds with a very high probability as soon as the database has a reasonable size. As a consequence, the full PSS protocol becomes insecure in practice.

After that, we show that all PMSP protocols, that are computed by using additive masking functions, must reveal the differences between the scalar products. We then propose two new SMSP protocols that do not reveal anything else, except these differences, in the security-model proposed by Du and Atallah (security against ciphertext-only attacks with the help of a semi-honest third party Ursula). This protocol is as efficient as the Du-Atallah PMSP protocol and quantifiably more secure.

However, ciphertext-only security is not sufficient in many practical applications, since additional public information about the database may easily disclose private information. We argue that practical PSS protocol must at least withstand statistical attacks (where the attacker knows something non-trivial about the database) and known-plaintext attacks. We show that any PMSP protocol, where Ursula learns distance differences, is insecure against very simple statistical attacks. While a refinement to the additive masking (the use of order-preserving affine transformation, defined later in this paper) can resist simple statistical attacks, it is not sufficient for more elaborate attacks. Finally, in Sect. IV, we propose the divide and conquer technique that provides more security, but it is also computationally more demanding.

We stress that also the protocols, proposed in this paper, are applicable only in limited environments, e.g., when only the ciphertext-only attacks are allowed. Still, the provably secure alternative of using Yao's alternative garbled circuit evaluation [4] is computationally too costly for large databases, and therefore our new protocols could be used in the practice

when a trade-off between security and efficiency is desired. It is a major open problem to design a PSS protocol that is both secure and efficient.

Road-map. Section II introduces the reader to notation and preliminaries. Section III describes our attacks against MINDASP protocol and improved PSS protocols. Section IV gives a brief overview of other possible attack scenarios and solutions.

II. Notation and Preliminaries

Notation. A PSS database can consist of either discrete, continuous or hybrid vectors. For example, genome data can be represented over discrete alphabet, whereas biometric data is inherently continuous. For discrete data, we use elements of some quotient ring $\mathbb{Z}_t$, and we think of continuous parameters as fixed point real numbers that are mapped into $\mathbb{Z}_t$ by using a suitable affine transform.

For n-dimensional vectors, two standard distance functions are the Euclidean distance $d_2(\boldsymbol{x},\boldsymbol{y}) := \left[\sum_i (x_i - y_i)^2\right]^{1/2}$ and the Manhattan distance $d_1(\boldsymbol{x},\boldsymbol{y}) := \sum_i |x_i - y_i|$, where all calculations are done in $\mathbb{R}$. (If made in $\mathbb{Z}_t$, no modular reductions should appear in the computations.)

Cryptographic background. A public-key cryptosystem Π is a triple (G, E, D) of probabilistic polynomial-time algorithms for key-generation, encryption and decryption. A cryptosystem Π is homomorphic if $E_K(m_1; r) \cdot E_K(m_2; s) = E_K(m_1 + m_2; r \cdot s)$, where $+$ is a group operation and $\cdot$ is a groupoid operation, and semantically secure if no probabilistic polynomial-time adversary can distinguish between random encryptions of two elements, chosen by herself. See, e.g., [5], for an efficient semantically secure homomorphic public-key cryptosystem.

Linear-algebraic background. The next results are necessary to quantify our attacks in the next section. For the sake of completeness, we present them together with proofs.

First, let us denote the ring of $m \times n$ matrices over the ring $\mathbb{Z}_t$ by $\mathsf{Mat}_{m\times n}(\mathbb{Z}_t)$. Recall that a matrix equation $M\boldsymbol{x} = \boldsymbol{y}$ over the finite field $\mathsf{GF}(q) = \mathbb{Z}_q$ can have at most $q^{n-\mathsf{rank}(M)}$ solutions. Generally, when $\mathbb{Z}_t$ is a ring, the solution of $M\boldsymbol{x} = \boldsymbol{y}$ is unique iff the matrix M is left-invertible. Let $\mathcal{L}$ be the set of left-invertible matrices. Define $\mathsf{P}_{m\times n}(t) := \Pr[M \leftarrow \mathsf{Mat}_{m\times n}(\mathbb{Z}_t) : M \in \mathcal{L}]$.

Lemma 1: Let $m \geq n$, let $q, q_1, \ldots, q_\ell$ be some primes. Then the following claims hold: (a) $\mathsf{P}_{m\times n}(q) = \prod_{k=0}^{n-1}\left(1 - q^{-m+k}\right)$;
(b) $\mathsf{P}_{m\times n}(q^r) = \mathsf{P}_{m\times n}(q)$;
(c) $\mathsf{P}_{m\times n}(q_1^{r_1}\cdots q_\ell^{r_\ell}) = \prod_{i=1}^{\ell}\mathsf{P}_{m\times n}(q_i)$.

Proof: (a) Since $\mathbb{Z}_q$ is a finite field, a matrix M over $\mathbb{Z}_q$ is left-invertible iff all columns of M are linearly independent. Now, if the first k columns of M are linearly independent, they span a vector space of dimension k and size q^k. The probability that the next column avoids this space is $1 - q^k/q^m$ and thus probability that all n columns are linearly independent is $\prod_{k=0}^{n-1}(1 - q^{-m+k})$.
(b) Follows directly since M has a left-inverse modulo q^r iff $M \mod q$ has a left-inverse modulo q. (c) By the Chinese remainder theorem, matrices $M_i \equiv M \mod q_i^{r_i}$ for every i have left-inverses iff M does. Random sampling in $\mathbb{Z}_t$ is equivalent to random sampling modulo $q_i^{r_i}$ for every i, and thus the probability in this case is just a product of the probabilities given by case (a). ∎

Private similarity search. A private similarity search (PSS) protocol has two parties, the querier Alice and the database owner Bob. Alice's private input is a vector (query) $\boldsymbol{x}$; Bob's private input is the database with vector elements $\boldsymbol{y_1}, \ldots, \boldsymbol{y_m}$. Assume that the similarity between two vectors is determined by a public distance (score) function $d(\cdot,\cdot)$. During a PSS protocol, Alice learns the *match index* b, s.t. $\boldsymbol{y_b} = \arg\min_{\boldsymbol{y_i}} d(\boldsymbol{x}, \boldsymbol{y_i})$. and the corresponding *match score* $d(\boldsymbol{x}, \boldsymbol{y_b}) = \min_i d(\boldsymbol{x}, \boldsymbol{y_i})$. If b is not unique, Bob may return a randomly chosen index that minimises b. Alice must gain no new information, except her private output (the match index and the match score). Bob must gain no new information. Some PSS protocols use a conditionally trusted third party Ursula, who must gain no new information during the protocol.

Du-Atallah protocol for finding minimal Euclidean distance. The task of finding the closest match can be simplified when $d(\cdot,\cdot)$ is the Euclidean distance. Then one can ignore the square root and compute $d^2(\boldsymbol{x}, \boldsymbol{y_i}) := \sum_{j=1}^{n}(x_j - y_{ij})^2$, where $\boldsymbol{y_i} = (y_{i1}, \ldots, y_{in})$. Moreover, since $d^2(\boldsymbol{x}, \boldsymbol{y_i}) = \boldsymbol{x}^2 - 2\boldsymbol{x}\cdot\boldsymbol{y_i} + \boldsymbol{y_i}^2$ and $\boldsymbol{x}^2$ is a constant known to Alice, it is sufficient for Alice to learn the minimal value of $-2\boldsymbol{x}\cdot\boldsymbol{y_i} + \boldsymbol{y_i}^2$ over all i-s. The latter task can be reduced to the private minimal scalar product problem by defining new vectors $\boldsymbol{x'} := (-2x_1, \ldots, -2x_n, 1)$ and $\boldsymbol{y'_i} := (y_{i1}, \ldots, y_{in}, \sum_{j=1}^{n} y_{ij}^2)$ for all i-s; then $\boldsymbol{x'}\cdot\boldsymbol{y'_i} = -2\boldsymbol{x}\cdot\boldsymbol{y_i} + \boldsymbol{y_i}^2$.

Since only generic inefficient protocols are known for the minimum finding [2], the hardest part of any reasonable PSS protocol is to find the minimum over the distances. To overcome this issue Du and Atallah proposed in [1] to use a trusted third party Ursula. Their proposed protocol, MINDASP, is depicted by Protocol 1.

INPUT: A query $\boldsymbol{x}$ and a database $\boldsymbol{y_1}, \ldots, \boldsymbol{y_m}$.
OUTPUT: The index b and the score $\boldsymbol{x}\cdot\boldsymbol{y_b}$.

1) Alice and Bob jointly generate two random numbers r^A, r^B.
2) For every row i:
   a) Alice and Bob jointly generate two random vectors $\boldsymbol{R_i^A}, \boldsymbol{R_i^B}$.
   b) Alice sends $\boldsymbol{w_i^A} \leftarrow \boldsymbol{x} + \boldsymbol{R_i^A}$ and $s_i^A \leftarrow \boldsymbol{x}\cdot\boldsymbol{R_i^B} + r^A$ to Ursula.
   c) Bob sends $\boldsymbol{w_i^B} \leftarrow \boldsymbol{y_i} + \boldsymbol{R_i^B}$ and $s_i^B \leftarrow \boldsymbol{R_i^A}\cdot\boldsymbol{w_i^B} + r^B$ to Ursula.
   d) Ursula computes and stores $v_i \leftarrow \boldsymbol{w_i^A}\cdot\boldsymbol{w_i^B} - s_i^A - s_i^B$.
3) Ursula finds an index b for which $v_b = \min_i v_i$. She sends b and v_b to Alice. Alice outputs $v_b + r^A + r^B$.

Protocol 1: MINDASP protocol.

During this protocol, for every i, Ursula learns the value $v_i = \boldsymbol{x}\cdot\boldsymbol{y_i} - r^A - r^B$ for r_A and r_B unknown to her. Therefore, provided that Ursula is honest, Alice learns the correct answer. Though Ursula gains new non-trivial information about the query, the authors of [1] argue that it is not sufficient to

reveal either the match scores, the queries or the database. However, [1] does not give any further analysis. It also does not specify how to choose random values. In the case of a discrete search space, we should use $\mathbb{Z}_t$ with t being larger than any intermediate scalar product. More precisely, let $\Delta := \max\{\boldsymbol{x} \cdot \boldsymbol{y_i} - \boldsymbol{x} \cdot \boldsymbol{y_j}\}$; then we must have $t > 2\Delta$ since otherwise, Ursula cannot determine the smallest v_i. Thus, in the discrete case, Alice and Bob first agree on a safe $\mathbb{Z}_t$, perform all calculations in $\mathbb{Z}_t$ and choose all necessary random numbers uniformly from $\mathbb{Z}_t$. If the vectors are continuous, Alice and Bob have to embed real numbers into large enough $\mathbb{Z}_t$. More precisely, they should first fix the precision parameter p and use transformation $x \mapsto \lfloor p \cdot x \rceil$. Therefore, we must have $t > p^2\Delta$, since otherwise Ursula cannot determine the minimum.

Du-Atallah security model. In the security model of Du and Atallah, only ciphertext-only attacks with the next restrictions, are allowed: Ursula colludes with nobody and has no prior information about queries and a database. A protocol is considered secure in this model if Ursula cannot restore any queries, database vectors or corresponding match scores. It is still required that Bob must learn no new information and Alice must only learn the match index and the match score. It is easy to see that during the MINDASP protocol, Ursula can trivially learn distance differences. This does not directly imply that ciphertext-only attacks are dangerous to this protocol in the Du-Atallah security model, as the differences themselves do not reveal any vectors or match scores.

III. ANALYSIS OF SMSP PROTOCOLS

Efficient ciphertext-only attack against MINDASP. Recall that m is the number of database elements and n is the dimension of the vectors.

Lemma 2: Assume that the MINDASP protocol is executed over the ring $\mathbb{Z}_t$. If $m > n$, a semi-honest Ursula can reconstruct $\boldsymbol{x}$ with probability $\mathsf{P}_{(m-1)\times n}(t)$.

Proof: For any i, $s_i^B = (\boldsymbol{w_i^A} - \boldsymbol{x}) \cdot \boldsymbol{w_i^B} + r^B$ and thus $\boldsymbol{w_i^B} \cdot \boldsymbol{x} = \boldsymbol{w_i^A} \cdot \boldsymbol{w_i^B} + r^B - s_i^B$. Hence, Ursula obtains $m-1$ equations $(\boldsymbol{w_i^B} - \boldsymbol{w_1^B}) \cdot \boldsymbol{x} = \boldsymbol{w_i^A} \cdot \boldsymbol{w_i^B} - \boldsymbol{w_1^A} \cdot \boldsymbol{w_1^B} - (s_i^B - s_1^B)$. The claim now follows from Lemma 1. ∎

As an example, if $t = 10$, $n = 6$ and $m = 7$, the success probability is roughly 0.22, while $m = 11$ raises it to 0.94. If $m > 2n$, the success rate will be almost 1.

Improved protocols. Next, we propose two alternative PMSP protocols that achieve the security goal, proposed by Du and Atallah: namely, Alice learns exactly the closest match and the match index, Bob gains no new information and Ursula learns only the distance differences $d_{ij} := d(\boldsymbol{x}, \boldsymbol{y_i}) - d(\boldsymbol{x}, \boldsymbol{y_j})$.

The MINTSP protocol, depicted by Protocol 2, is a simple tweak to MINDASP that achieves the required security level. (Note that scalar products in the MINTSP protocol represent distances up to an additive constant.)

Lemma 3: Assume the participants are semi-honest. During a single run of Protocol 2, Alice learns only the match score and match index, Bob learns nothing and Ursula learns only the scalar product differences $d_{ij} := \boldsymbol{x} \cdot \boldsymbol{y_i} - \boldsymbol{x} \cdot \boldsymbol{y_j}$.

INPUT: A query $\boldsymbol{x}$ and a database $\boldsymbol{y_1}, \ldots, \boldsymbol{y_m}$.
OUTPUT: The index b and the score $\boldsymbol{x} \cdot \boldsymbol{y_b}$.

1) Alice and Bob jointly a random query key r.
2) For every row i:
   a) Alice and Bob randomly choose vectors $\boldsymbol{R_i^A}, \boldsymbol{R_i^B}$ and a scalar r_i.
   b) Alice sends $\boldsymbol{w_i^A} \leftarrow \boldsymbol{x} + \boldsymbol{R_i^A}$ and $s_i^A \leftarrow \boldsymbol{x} \cdot \boldsymbol{R_i^B} + r_i$ to Ursula.
   c) Bob sends $\boldsymbol{w_i^B} \leftarrow \boldsymbol{y_i} + \boldsymbol{R_i^B}$ and $s_i^B \leftarrow \boldsymbol{R_i^A} \cdot \boldsymbol{w_i^B} - r - r_i$ to Ursula.
   d) Ursula computes and stores $v_i \leftarrow \boldsymbol{w_i^A} \cdot \boldsymbol{w_i^B} - s_i^A - s_i^B$.
3) Ursula finds the minimising index b such that $v_b = \min_i v_i$. Then sends b and v_b to Alice, who finds the score $v_b - r$.

Protocol 2: The MINTSP protocol

Proof: Correctness is clear, since $\boldsymbol{w_i^A} \cdot \boldsymbol{w_i^B} - s_i^A - s_i^B = (\boldsymbol{x} + \boldsymbol{R_i^A}) \cdot (\boldsymbol{y_i} + \boldsymbol{R_i^B}) - (\boldsymbol{x} \cdot \boldsymbol{R_i^B} + r_i) - (\boldsymbol{R_i^A} \cdot (\boldsymbol{y_i} + \boldsymbol{R_i^B}) - r - r_i) = \boldsymbol{x} \cdot \boldsymbol{y_i} + r$. It is straightforward to simulate the views of Alice and Bob, and therefore nothing will be leaked to them. To prove that nothing else, except the values d_{ij}, is leaked to Ursula, we show how to perfectly simulate views of Ursula by a simulator who knows all the differences d_{ij}. Consider the view of Ursula. In a valid protocol run, Ursula sees tuples $(\boldsymbol{w_i^A}, \boldsymbol{w_i^B}, s_i^A, s_i^B)$, such that $\boldsymbol{w_i^A} \cdot \boldsymbol{w_i^B} - s_i^A - s_i^B = \boldsymbol{x} \cdot \boldsymbol{y_i} + r$. Since $\boldsymbol{R_i^A}, \boldsymbol{R_i^B}, r_i$ are chosen uniformly, the triple $(\boldsymbol{w_i^A}, \boldsymbol{w_i^B}, s_i^A)$ has also a uniform distribution. Consequently, the simulator can choose $\bar{v}_1, \bar{\boldsymbol{w}}_{\boldsymbol{i}}^{\boldsymbol{A}}, \bar{\boldsymbol{w}}_{\boldsymbol{i}}^{\boldsymbol{B}}, \bar{s}_i^A$ at random and compute $\bar{s}_i^B = \bar{\boldsymbol{w}}_{\boldsymbol{i}}^{\boldsymbol{A}} \cdot \bar{\boldsymbol{w}}_{\boldsymbol{i}}^{\boldsymbol{B}} - \bar{s}_i^A - \bar{v}_1 - d_{i1}$. ∎

To reduce the number of required random bits and the communication, Alice and Bob can use a pseudo-random generator for generating $r, \boldsymbol{R_i^A}, \boldsymbol{R_i^B}, r_i$. Then they have to agree only on random seed s and only send the tuple $(\boldsymbol{w_i^A}, \boldsymbol{w_i^B}, s_i^A, s_i^B)$ to Ursula. This reduces the communication between Alice and Bob to a few hundred bits.

Provided that vectors belong or can be safely embedded into $\mathbb{Z}_t^n$, where $\mathbb{Z}_t$ is the plaintext space of the used semantically secure homomorphic public-key cryptosystem $\Pi = (G, E, D)$, one can alternatively use the next communication-efficient MINHSP protocol, depicted by Protocol 3. Note that the same key K can be used in multiple protocols.

ALICE'S INPUT: A query $\boldsymbol{x} = (x_1, \ldots, x_n)$
BOB'S INPUT: A database $\boldsymbol{y_1}, \ldots, \boldsymbol{y_m}$, where $\boldsymbol{y_i} = (y_{i1}, \ldots, y_{in})$.
OUTPUT: The index b and the score $\boldsymbol{x} \cdot \boldsymbol{y_b}$.

1) Ursula generates a new private-public key pair for Π and sends the public key K to Alice and Bob.
2) Alice and Bob choose a random r from the plaintext space of Π.
3) For each $j \in [n]$, Alice sends $c_j \leftarrow E_K(x_j; t_j)$ to Bob, where t_j is a fresh random number.
4) For each row i, Bob sends $s_i \leftarrow \prod c_j^{y_{ij}} \cdot E_K(r; t)$, for a fresh random number t, to Ursula.
5) Ursula decrypts the result and sends the match index b and $v_b \leftarrow D_K(s_b)$ to Alice. Alice computes the score $v_b - r$.

Protocol 3: The MINHSP protocol

Alice's and Bob's privacy in the MINHSP protocol relies on the semantical security of Π. The security proof for Protocol 3

is very standard and therefore omitted. The MINHSP protocol requires n encryptions by Alice, and mn exponentiations by Bob. The number of exponentiations can be amortised. For example, if vectors $\boldsymbol{y_i}$ consist of binary data, Bob will not have to perform any exponentiations. Hence, the most demanding computational burden of m decryptions is placed on Ursula. The communication of the MINHSP protocol is only $m+n$ ciphertexts, whereas the MINTSP protocol requires sending at least $2m(n+1)$ scalars. Since the MINTSP protocol is computationally more efficient, we will have a trade-off between communicational and computational complexity.

IV. SECURITY AGAINST MORE ELABORATED ATTACKS

To avoid the costly minimum finding operation, the previous protocols make use of a trusted third party Ursula. However, Ursula learns some non-trivial information—namely, the distance differences—about the queries and the database. Next, we will analyse how Ursula can abuse this information and what could be the possible counter-measures.

Known-plaintext attacks. As Ursula obtains the list of distance differences $d_{ij} = \boldsymbol{x}\cdot\boldsymbol{y_i} - \boldsymbol{x}\cdot\boldsymbol{y_j}$, the knowledge of $\boldsymbol{x}\cdot\boldsymbol{y_i}$ for any single i reveals all scalar products. If Ursula knows $r > n$ database elements $\boldsymbol{y_{i_0}}, \ldots, \boldsymbol{y_{i_r}}$, she will obtain r equations $d_{i_k i_0} = \boldsymbol{x}\cdot(\boldsymbol{y_{i_k}} - \boldsymbol{y_{i_0}})$. Consequently, she can restore all query vectors $\boldsymbol{x}$, provided that $\boldsymbol{y_{i_1}} - \boldsymbol{y_{i_0}}, \ldots, \boldsymbol{y_{i_r}} - \boldsymbol{y_{i_0}}$ are linearly independent. The latter holds for a random database with probability, given by Lemma 1. By a similar argument, the knowledge of $r > n$ linearly independent query vectors to the same database reveals all differences $\boldsymbol{y_j} - \boldsymbol{y_1}$.

Some simple attacks can be avoided by randomly permuting the database elements before each query. This forces Ursula to determine the values $v_{j_0}, \ldots, v_{j_r}$ that are paired with $\boldsymbol{y_{i_0}}, \ldots, \boldsymbol{y_{i_r}}$. Since the number of valid pairings is $m(m-1)\cdots(m-r+1)$, where m is the number of database elements, such attacks become infeasible for most databases, at least if assuming that Ursula does not have any extra knowledge about the database.

Statistical attacks. However, the random permuting of the database rows does not provide absolute protection since Ursula still learns the multi-set $\{d(\boldsymbol{x}, \boldsymbol{y_i}) + r\}$. If n is large and the database vectors contain enough entropy, then one can approximate the empirical distance distribution with a data-independent distribution (this is caused by the *curse of high dimensions*). This statement is of course purely qualitative; quantitative estimates require the knowledge of the distribution of query and database vectors. For example, if the database and query vectors are uniformly and at random chosen from $\{0,1\}^n$, then the distribution of d_2^2 can be approximated with the Gaussian distribution $\mathcal{N}(\frac{n}{2}, \frac{n}{4})$. Similar results can be obtained if the vector components are assumed to be (weakly) independent; then the desired results follow from the theorems of weak convergence.

Assume now that the distance distribution $\mathcal{D}$ is known to Ursula and for a single query, all distances are independently sampled from $\mathcal{D}$. Consequently, Ursula can compute several point estimators like the mean value, the median or the variance and then use the obtained knowledge to reconstruct the match score. For example, in the case of the MINTSP and the MINHSP protocols, one can compute the expected value of v_i, $\mathsf{Exp}(v_i) := \frac{1}{m}\sum_{i=1}^{m} v_i = \mathsf{Exp}(d_i) + r$. Therefore, $d(\boldsymbol{x}, \boldsymbol{y_b}) = v_b - \mathsf{Exp}(v_i) + \mathsf{Exp}(d_i)$. Since by assumption, all d_i-s are sampled randomly from $\mathcal{D}$, then the standard central limit theorem assures that $\mathsf{Exp}(d_i)$ has the Gaussian distribution $\mathcal{N}(\mu, \sigma^2/m)$, where μ and σ are the mean and the variance of the distribution $\mathcal{D}$. Let $d_* = v_b - \mathsf{Exp}(v_i) + \mu$, then one can use standard results from statistics to show that the match score $d(\boldsymbol{x}, \boldsymbol{y_b})$ is in the interval $d_* \pm \sigma^2/\sqrt{m}$ with the probability 68%. For example, if $n = 100$, $\mu = 50$ and $m > \sigma^2 = 625$, Ursula can with probability 68% infer the match score with precision ± 1. Ursula can estimate the variance $\sigma^2 \approx \mathsf{Exp}((v_i - \mathsf{Exp}(v_i))^2) = 1/m\sum_i (v_i - \mathsf{Exp}(v_i))^2$ directly from the multi-set $\{d(\boldsymbol{x}, \boldsymbol{y_i}) + r\}$ and thus also compare the different match scores without knowing the distribution.

Detection of identical queries. In all presented protocols, Ursula can detect identical queries, since identical (or even just similar) queries have identical (resp., similar) distance multi-sets $\{d(\boldsymbol{x}, \boldsymbol{y_i})\}$. Thus, with a high probability, Ursula can decide whether two queries were equal or not. The rate of false positives—two different queries that have have identical or similar distance multi-sets—depends on database vectors, but is certainly small. E.g., this rate can be computed if all distances are sampled independently from $\mathcal{D}$. If the queries are similar, $d(\boldsymbol{x_1}, \boldsymbol{x_2}) < \tau$, then $|d(\boldsymbol{x_1}, \boldsymbol{y_i}) - d(\boldsymbol{x_2}, \boldsymbol{y_i})| < \tau$. If v_i^1 and v_i^2 are ordered lists of the values v_i obtained in the PSS protocols then for similar queries, $|v_i^1 - v_i^2 - v_1^1 + v_1^2| < \tau$.

Order preserving transformations. The previously described protocols make use of the simplest order preserving transformation $d_i \mapsto d_i + r$. This map corresponds to the use of one-time pad, and thus the knowledge of a single distance compromises all distances. If say a randomised affine transformation $d_i \mapsto s d_i + r + \epsilon$, with a random distortion ϵ, is used, we get a bit more security but the database can still be completely determined by a known distance pair.

In the MINTSP and MINHSP protocols, it is straightforward to replace the additive masking function with an affine transformation. First note that not all affine transformations preserve order: (a) $\Delta := \max\{\boldsymbol{x}\cdot\boldsymbol{y_i} - \boldsymbol{x}\cdot\boldsymbol{y_j}\} < t/(2s)$ must hold to avoid modular reduction; (b) $0 < \epsilon < s$ must hold to avoid local re-ordering. Thus, to implement affine masking, Alice must choose s randomly from valid interval $s \in [s_{min}, s_{max}]$, where s_{min} and s_{max} are chosen so that the resulting transformation would preserve order, and then use $s\boldsymbol{x}$ instead of $\boldsymbol{x}$, while Bob must use $r + \epsilon$ instead of r. However, as not all values of s are valid, Ursula knows that $d_i - d_j \in [(v_i - v_j)/s_{max}, (v_i - v_j)/s_{min}]$. Hence, the ratio s_{max}/s_{min} classifies the maximum uncertainty, though the knowledge of Δ sometimes allows to exclude part of the interval. Therefore, the knowledge of the mean value μ of $\mathcal{D}$ does not reveal the match score if $s_{max}/s_{min} > \mu/(\mu - d_{min})$, where $d_{min} = \min_i d_i$.

Divide-and-conquer technique. The underlying idea of divide-and-conquer approach (see Protocol 4) is to find mini-

mum in several stages with a tournament scheme.

INPUT: A query $\boldsymbol{x}$ and a database $\boldsymbol{y_1}, \ldots, \boldsymbol{y_m}$.
OUTPUT: The index b and the score $\boldsymbol{x} \cdot \boldsymbol{y_b}$.
INITIAL STAGE

1) Bob randomly permutes the database and divides it into k random almost equally-sized blocks $\boldsymbol{y^{(1)}}, \ldots, \boldsymbol{y^{(k)}}$.
2) For every $j \in \{1, \ldots, k\}$:
   a) Run a minimal scalar product protocol on inputs $\boldsymbol{x}$ and $\boldsymbol{y^{(j)}}$, so that Ursula obtains the match index b_j and $v_{b_j} \leftarrow \min_i \boldsymbol{x} \cdot \boldsymbol{y_i^{(j)}}$.
   b) Ursula generates two random values $b_j^B \mod k$ and $d_j^B \mod t$, and sends them to Bob. She sends $b_j^A \leftarrow b_j - b_j^B \mod k$ and $d_j^A \leftarrow v_{b_j} - d_j^B \mod t$ to Alice.

SECOND STAGE

1) Alice and Bob jointly generate a random permutation π.
2) Alice and Bob choose a random key r.
3) For every index j, Ursula learns $v_{\pi(j)} = d^A_{\pi(j)} + d^B_{\pi(j)} + r$.
4) Ursula sends minimising index $\tau := \arg\min_j v_{\pi(j)}$ to Alice.
5) Alice uses private information retrieval to get d_τ^B and b_τ^B and compute d_b and b.

Protocol 4: Divide and conquer algorithm

As the minima are taken over smaller k-element blocks, the scheme is more resistant against statistical attacks and the empirical point estimates are less precise. It also makes harder to detect identical (similar) queries, since Ursula sees only a small random fraction shares $v_{i_1}, \ldots, v_{i_k}$ in every stage.

It is easy to generalise this protocol to an arbitrary number of stages. Again, instead of the additive masking, Alice and Bob can use more complex order-preserving transformations, for example affine transformation. Consider for example the extreme case, where there are only two values in each block. Then statistical attacks are not applicable and the distance difference is bounded to interval $[(v_1 - v_2)/s_{max}, (v_1 - v_2)/s_{min}]$. Moreover, Ursula cannot link different stages—for each comparison there are two equiprobable ways to assign winner. Consequently, it is infeasible to look through all possible tournament trees. Therefore, the scheme is secure against statistical attacks.

This protocol is computationally more demanding since computationally-private information retrieval protocols are not cheap computationally or communicationally. (The currently most communication-efficient protocol [6] has communication complexity $\Theta(\log^2 n \cdot k + \log n \cdot \ell)$, where n is the size of the database, k is the security parameter and ℓ is the bit-length of transferred strings.) Still, it is several orders of magnitude more efficient than Yao's garbled circuit evaluation [2]. The latter requires roughly one oblivious transfer per input bit and a large garbled circuit description.

Attacks against some other PSS protocols from [1]. Random permutations are not always applicable and thus known-plaintext attacks can be dangerous. For example, Du and Atallah proposed two protocols (see sections 4.4.1 and 4.5.1 from [1]) for PSS for Euclidean distance, when the database itself is outsourced to potentially hostile party. In both cases, Alice outsources her database $\boldsymbol{y_1}, \ldots, \boldsymbol{y_m}$ in a garbled form to Bob. Omitting unnecessary details, Bob receives $\boldsymbol{z_i} = Q\boldsymbol{y_i'}$ where Q is a random invertible matrix known only to Alice. Each query vector is in form $\boldsymbol{q} = Q^{-1}\boldsymbol{x'}$ so that Bob can compute $v_i = \boldsymbol{q} \cdot \boldsymbol{z_i} = \boldsymbol{x'} \cdot \boldsymbol{y_i'}$.

In the SSO protocol $\boldsymbol{x'}$ and $\boldsymbol{y_i'}$ is chosen so that $v_i = (\boldsymbol{x} - \boldsymbol{y_i})^2 + r$ for all i-s. Therefore, the leakage of $\boldsymbol{y_{i_0}}, \ldots, \boldsymbol{y_{i_r}}$ has a devastating effect. Bob can solve the linear equations $v_{i_k} - v_{i_0} + \boldsymbol{y_{i_0}}^2 - \boldsymbol{y_{i_k}}^2 = -2\boldsymbol{x} \cdot (\boldsymbol{y_{i_k}} - \boldsymbol{y_{i_0}})$ and determine query vectors. Then he can use revealed query vectors $\boldsymbol{x_1}, \ldots, \boldsymbol{x_s}$ to reconstruct the database.

In the SSCO protocol, Bob obtains $v_i = s(\boldsymbol{x} - \boldsymbol{y_j})^2 + r$. Thus Bob can treat equality $v_{i_k} - v_{i_0} = -2s\boldsymbol{x} \cdot (\boldsymbol{y_{i_k}} - \boldsymbol{y_{i_0}}) + s(\boldsymbol{y_{i_k}}^2 - \boldsymbol{y_{i_0}}^2)$ as linear equation with unknowns $\boldsymbol{x_*} = s\boldsymbol{x}$ and s. Similarly, the revealed query vectors $\boldsymbol{x_1}, \ldots, \boldsymbol{x_k}$ enable to reconstruct the database. To conclude, both protocols are insecure in practice—only a small leakage of database vectors compromises the entire garbled database.

Acknowledgements: This work was partially supported by the Estonian Information Technology Foundation, the Finnish Defence Forces Research Institute of Technology and by the Finnish Academy of Sciences.

REFERENCES

[1] W. Du and M. J. Atallah, *Protocols for Secure Remote Database Access with Approximate Matching*, ser. Advances in Information Security. Kluwer Academic Publishers, Boston, 2001, vol. 2, p. 192, http://www.wkap.nl/prod/b/0-7923-7399-5.

[2] M. Naor, B. Pinkas, and R. Sumner, "Privacy Preserving Auctions and Mechanism Design," in *The 1st ACM Conference on Electronic Commerce*, Denver, Colorado, Nov. 1999.

[3] O. Goldreich, *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[4] A. C.-C. Yao, "Protocols for Secure Computations (Extended Abstract)," in *23rd Annual Symposium on Foundations of Computer Science*. Chicago, Illinois, USA: IEEE Computer Society Press, 3–5 Nov. 1982, pp. 160–164.

[5] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Advances in Cryptology — EUROCRYPT '99*, ser. Lecture Notes in Computer Science, J. Stern, Ed., vol. 1592. Prague, Czech Republic: Springer-Verlag, 2–6 May 1999, pp. 223–238.

[6] H. Lipmaa, "An Oblivious Transfer Protocol with Log-Squared Total Communication," International Association for Cryptologic Research, Tech. Rep. 2004/063, Feb. 25 2004. [Online]. Available: http://eprint.iacr.org/2004/063/

Carnival
An Application Framework for Enforcement of Privacy Policies

Ragni R. Arnesen, Jerker Danielsson, and Bjørn Nordlund

***Abstract*—This paper presents Carnival, a framework providing privacy access control and audit functionality to application developers. Carnival enforces privacy policies that regulate access based on the action requested, the identity and/or roles of the requesting user, the purpose of the access, the identity and preferences of the data subject associated with the data, and the type of personal data to be accessed. A description of our implementation of Carnival in Java, and an outline of how to develop and deploy applications using this framework, is provided.**

***Index Terms*—Access control, Privacy, Privacy policy enforcement.**

I. Introduction

The presence of extensive data collection and processing capabilities threatens the privacy of individuals and organizations. However, the inherent privacy intruding effect of these collection and processing practices can be substantially reduced.

Data collectors should analyze their current collection practices and evaluate the types and amount of data collected, whether the collection is "needed and worth it" and whether pseudonymized data or less granular data is sufficient for the purposes of the data collection. (See e.g. Hansen and Pfitzmann [11] for a definition of pseudonymity.) Furthermore, under some circumstances it is possible to let the data subject, i.e. the person whose identity is, or may be, connected to the data, remain in control over her personal data by letting her control the transformation between pseudonym and identifier. That is, the data subject controls the keys that unlock the pseudonyms.

However, there are circumstances where processing of identifiable personal data is both useful and necessary. For example, medical data must be collected and processed within the hospital sector, and banks need personal data to evaluate customers' credit. In other cases, processing of personal data (possibly pseudonymized) is not strictly necessary, but may be of benefit to both data collector and data subject. An example of such a case is the possibility for a data collector to customize offers to the data subject based on her interests, history, and current context, e.g. location.

In any case, as long as the personal data is not anonymized, its use needs to be regulated. This paper proposes an automated mechanism for mandatory enforcement of privacy promises given to customers.

A. Motivation

The data subject whose personnel data is collected and stored usually has little control over its usage. The notion of privacy when personal data is collected implies some form of trust in the data-collecting entity, but this trust is not necessarily extended to its employees. There is thus a need for privacy protection mechanisms to enforce the privacy promises made by data-collecting organizations to data subjects (e.g. customers).

Furthermore, privacy is very subjective. Different people have different opinions of what is privacy intrusive and what is not, and also on whether an entity is trustworthy or not. In other words, people have different privacy preferences, and should be allowed to express these preferences and have them respected. For an organization having thousands of customers with different privacy preferences, automated solutions for privacy enforcement are necessary.

A system for automated and mandatory enforcement of privacy policies would provide a tool for organizations to enforce the privacy promises given to customers. The implementation of such a system in an organization may contribute to the establishment of trust, and allow individual preferences to be taken into account.

This paper presents Carnival, a framework which, when integrated with applications, provides both access control regulated by privacy policies, and audit functionality to ensure accountability.

Carnival provides functionality for proactive and reactive control to ensure that the purpose of each access corresponds to the purpose stated when the data was collected. It provides a tool for organizations to ensure that their privacy promises are enforced and not breached by individuals associated with the organization.

B. Outline

Section II presents some related work in the area of privacy

Manuscript submitted August 2, 2004. The work presented in this paper is fully funded by the Norwegian Research Council through the research project "Personalized Internet-Based Services and Privacy Protection."

R. R. Arnesen is with Norsk Regnesentral (Norwegian Computing Center), P.O.Box 114 Blindern, NO-0314 Oslo, Norway (phone: +47 22852565; fax: +47 22697660; e-mail: Ragni.Ryvold.Arnesen@nr.no).

J. Danielsson, is with Norsk Regnesentral (Norwegian Computing Center), P.O.Box 114 Blindern, NO-0314 Oslo, Norway (e-mail: Jerker.Danielsson@nr.no).

B. Nordlund is with Norsk Regnesentral (Norwegian Computing Center), P.O.Box 114 Blindern, NO-0314 Oslo, Norway (e-mail: Bjorn.Nordlund@nr.no).

access control. How privacy access control differs from "traditional" access control is discussed in section III. Then, in section IV the functionality needed in a framework like Carnival is explored. How this functionality is provided by Carnival is discussed in section V, and section VI explains how Carnival is configured and used. Finally, section VII provides some closing remarks.

II. Related work

In [7] Fischer-Hübner presents a formal task-based privacy model for enforcement of privacy policies and its implementation. The central idea is to control access to personal data through strict control of the tasks users perform. In this setting, a task consists of a set of allowed transformation procedures. Access to personal data is only granted if it is necessary for the task, the user is authorized for the task, and the purpose of the task corresponds to the purpose stated when the information was collected, unless the user has consented to the new purpose.

Karjoth and Schunter present a privacy policy model for enterprises in [10]. They create a privacy control language that includes, among others, user consent, other conditions, and obligations. The policy model allows administration of the system authorizations to be distributed e.g. between a privacy officer and a security officer, while guaranteeing separation of duty.

IBM has developed Declarative Privacy Monitoring (DPM) [5]. DPM is a Java library for adding privacy access control and auditing functionality to J2EE web applications, and hence it can only be applied to applications running in a J2EE context. In contrast, our implementation of Carnival works with plain Java applications. Like Carnival, DPM provides access control based on the action requested, the identity/role of the requesting user, the purpose of the access, the identity and preferences of the data subject associated with the data, and the type of personal data to be accessed. However, DPM does not include any functionality for communicating the current task (i.e. purpose) to the user or functionality for the user to override the current task.

In [2] we present a framework for enforcement of privacy policies. Here we give a description of the functionalities that are necessary to enforce privacy policies and legislation. In the context of this framework Carnival implements the Reference Monitor and it provides a tool for generating the logs that are analyzed by the components of the Monitoring element.

The Hippocratic Database concept is introduced in [1]. It is argued that future database systems should include functionality for protecting the privacy of the data they store. A strawman design for such a database is presented. The design outlines, among others, how queries on the data in the database is regulated according to policy and how information about performed queries are logged.

One main difference between the proposed solutions is at which layer the privacy access control logic is applied. The purpose of an access is easiest determined at the layers closest to the user, whereas the personal data accessed is easiest determined at lower layers. DPM, like Carnival, implements access control in the data layer of the application and determines the purpose of access in higher layers. The Hippocratic Database implements access control in the database layer, and the implementation of Fischer-Hübner's privacy model implements access control in the operating system layer. These two last solutions require applications to propagate the purpose of accesses to the database and the operating system, respectively.

III. Privacy access control

Access control forms a necessary basis for enforcement of privacy, but it is important to realize that *privacy access control* is different from "traditional" access control. This is mainly for two reasons.

First, the *purpose* of data access is important. When personal information is collected, the purpose of the collection must be stated. If a subsequent request for access to the information is made, the purpose of the information access must correspond to the purpose stated when the information was collected. Using the information for other purposes should not be allowed unless the data subject consents, or there is a legal right (or obligation) to do this. These principles can be found in the OECD guidelines [12], and are important in most enacted privacy legislations (e.g. EU Directive [6]). The stated purpose of accesses is up to the discretion of the user and therefore audit is necessary to detect misuse through false purpose statements.

Second, access to personal information could lead to *obligations* that must be addressed. For example, legislation may require that a person should be notified when someone runs a credit check on him, or one may be required to delete or depersonalize information after a given period of time. In many cases, it is not possible to check that the obligations are fulfilled before information access is granted. Hence, proper workflow control and audit of system behavior are crucial to ensure that obligations are indeed fulfilled as required.

Privacy access control is regulated by the rules of a privacy policy. An example of such a rule written in plain English is: "An insurance agent (*role*) may read (*action*) information about my financial situation and living conditions (*data type*) if he uses it to offer me a tailored insurance package (*purpose*), provided that I am notified (*obligation*)". Such rules may be written in a machine-readable policy language, e.g. EPAL [3], for automated evaluation by a rule engine.

Carnival regulates access based on the current purpose of the user. Privacy policies, and the purpose statements they contain, may be rather abstract to be manageable and accessible to humans. Computer applications are generally only aware of what the user wants to do (i.e. the requested operation), not why (i.e. for which purpose). To automatically enforce abstract policies the stated purposes may have to be refined into more concretely defined purposes and these

purposes can be associated with the operations of the application.

If individual preferences are to be taken into account, there will be one set of policy rules for each individual in addition to the organization's policy. Thus, the identity of the data subject whose data is requested must be taken into consideration when determining which policy rules to evaluate against the access request. In addition, there may be a need to retrieve and evaluate different types of context information, such as access history, time of day, current location of the data subject, or whether or not a specific relation exists between the user and the data subject. This contributes to the complexity of implementing privacy access control.

IV. Requirements

This section explores some important requirements that apply to privacy access control mechanisms. The subsequent sections describe how Carnival meets these requirements.

To be able to evaluate access requests against a privacy policy, the following information must be retrieved for each access request:

- The identity and/or roles of the user who wants to access the data.
- The action requested on the data.
- The purpose of the access.
- The type of data requested.
- The identity of the data subject.

The identity of the data subject is needed to identify the data subject's individual policy, which formalizes the user's choices, consents and conditions. Note that this identity may be a pseudonym.

Additionally, it may be necessary to provide other information to evaluate deployment specific conditions. For example, the policy of a pharmacy might state that a pharmacist may only access prescriptions if the data subject of the prescription is present (e.g. proven by inserting a smart card into a reader). In this case the location of the data subject is also needed to evaluate access requests.

The access control mechanism must obviously include logic, or connect to logic, for evaluating access requests based on the information above. This evaluation logic should be easily replaceable; it should be possible to plug in evaluation logic implementing different access control models and supporting different policy languages.

Further, plug-ins for executing different types of obligations should be supported. Obligations that should be supported are obfuscation (e.g. making the data less granular) and pseudonymization of data before access is granted.

In addition, the access control mechanism should guarantee that the user and access control mechanism have the same understanding of what the user's purpose is. It is important to ensure that a user cannot make the case that he or she was accessing the data for another purpose than the one registered by the access control mechanism.

Finally, to ensure flexibility, the policy-based access control mechanism should be kept separate from the application code. The access control mechanism should not make any assumptions that the application in any way restricts access to personal data.

V. Carnival

Carnival intervenes in the execution of the application when users access personal data. The central privacy-enforcing element of Carnival is the Privacy Manager that protects data objects containing personal data in the application. In Carnival terminology such objects are called personal data objects.

The Privacy Manger can be seen as a container. Data in this container is privacy protected; data outside this container is not. Objects inside this container are called privacy-managed personal data objects (or just managed objects).

The Privacy Manager intercepts both before and after access to the personal data in managed objects. Before access, logging and access control is performed. After access, logging and obligation execution is performed. The Privacy Manager can be integrated with the application using a number of different techniques. In the current version of Carnival, Dynamic Proxies[1] are used. Another alternative could have been to use Aspects[2].

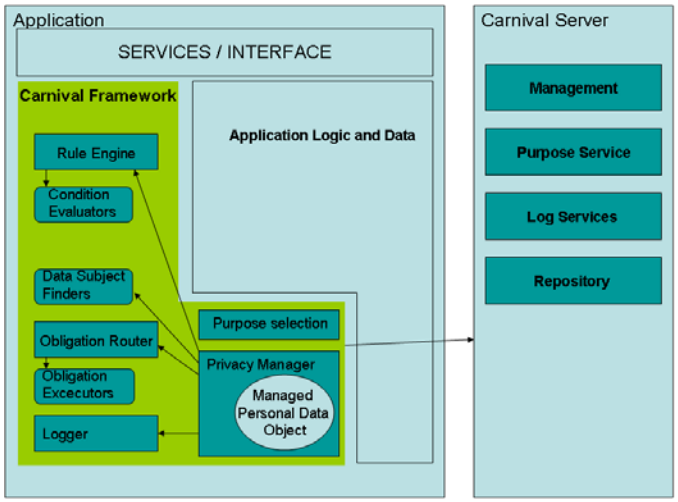


Figure 1 Overview of Carnival

A. Architecture

Carnival consists of the Carnival Framework and the Carnival Server (see Figure 1). The Carnival Framework is integrated into applications and it uses the services of the Carnival Server in the enforcement of the privacy policy of the organization.

The Carnival Framework is made up of the Privacy Manager, a number of services (rectangles in Figure 1), and optional developer- and deployer-supplied plug-ins (rounded corners in Figure 1).

[1] See http://java.sun.com/j2se/1.3/docs/guide/reflection/proxy.html
[2] See http://en.wikipedia.org/wiki/Aspect-oriented_programming

The Privacy Manager intercepts and collects information about access requests to personal data in managed objects. The Privacy Manager uses Data Subject Finder plug-ins to retrieve the identity of the data subjects whose personal data is requested.

The Rule Engine evaluates access requests on behalf of the Privacy Manager. The Rule Engine used by the current version of Carnival evaluates EPAL policies. It is possible to replace this Rule Engine with other implementation possibly supporting other access control models (e.g. Chinese Wall [4]) and/or policy languages. In the work of evaluating an access request the Rule Engine may call one or several Condition Evaluator plug-ins.

Accesses may result in obligations. These obligations are interpreted by the Obligation Router and handed off to the appropriate Obligation Executor plug-in.

Audit logs are created by the Logger service. It receives information about access requests and accesses to managed personal data objects from the Privacy Manager and constructs log records according to the log policy.

Finally, the Purpose Selection service implements logic for determining users' current purposes. A method for determining a user's current purpose is presented in section V.E.

The Carnival Server consists of:

- A Management interface, for managing organization policy (vocabularies, privacy policies, and log policies) and application configuration (links to user directory, purpose rules, and privacy metadata descriptor).
- A Repository, providing the Carnival Framework access to configuration and policy.
- A Purpose service, which stores users' current purposes. A central Purpose service enables the purposes of users to be determined based on their actions in different applications.
- A Log service, which receives and accumulates the logs created by the Logger.

Carnival enforces privacy policies that regulate access based on the action requested, the identity and/or roles of the requesting user, the purpose of the access, the identity and preferences of the data subject associated with the data, and the type of personal data to be accessed. This information is application independent. Hence, unfavorable coupling of policy and applications is avoided. The same organization-wide policy can be applied to all applications without any adaptation to the policy.

The Carnival Server leverages this decoupling between policy and applications by providing central management and integration of policy enforcement in applications. It offers a default implementation of the services that are needed by the Carnival Framework. This reference implementation may be replaced with other solutions implementing the interfaces and services required by the Carnival Framework.

B. Execution flow

Carnival regulates access to get and set methods[3] in managed personal data objects. Carnival requires that all access to personal data contained in personal data objects goes through get and set methods.

When data in a managed object is requested the Privacy Manager derives the action requested from the method called and retrieves the purpose and roles of the requesting user from the Carnival Server.

The Privacy Manager also retrieves information about the data subject and the data types of the data to be accessed. This information is collectively termed privacy metadata.

All this information is passed on to the Rule Engine that determines which policy to evaluate the request against. If an individual policy is available, it is used. Otherwise, the default local policy is used.

If the policy contains conditions these are evaluated by Condition Evaluator plug-ins. Condition Evaluators receive information about the access request (user, data subject, etc) from the Rule Engine. If the Condition Evaluator needs other information for evaluating the condition the Condition Evaluator retrieves this information from the application or some external information source.

The Rule Engine may decide that the user is denied access or, alternatively, that the access is granted. If access is denied, an exception is thrown, which the application should take appropriate actions to handle, e.g. roll back transactions and/or provide a message to the user.

The Rule Engine may associate the grant to access with one or several obligations, as determined by the applicable policy. If so, the obligations are passed on to the Obligation Router by the Privacy Manager. The Obligation Router routes the individual obligations to the correct Obligation Executor instances.

There are two types of Obligation Executors: synchronous and asynchronous. Synchronous Obligation Executors block until a result is returned. The Rule Engine may, for example, demand that, before access is granted, the level of detail in the result should be reduced according to a specification provided by the policy. For example, the age of the data subject may be replaced by an interval.

Asynchronous Obligation Executors do their job in the background. An example of such an Obligation Executor is one that is capable of sending notifications to data subjects through email.

The rest of this paper focuses on the access control functionality of Carnival. Under the hood the logging functionality is implemented much the same way as the access control functionality. The main difference is that in the case of logging, information collected is used to create a log record that is sent to the Log Service, whereas for access control the information is sent to the Rule Engine for evaluation. It is important that the log is subjected to manual and possibly

[3] A get method (e.g. getId) of an object retrieves the value of an instance variable (id) of the object. A set method modifies the value of an instance variable of an object.

automated audit to detect privacy violations.

C. Privacy metadata

The type of data to be accessed and whom the data is about are natural to extract from the objects in the application that represent the data subjects and that consequently contain information pertaining to data subjects. For example, in an Electronic Patient Journal (EPR) application it is natural to extract this metadata from the objects in the application that represent patients.

Consequently, Carnival introduces the concept of personal data classes and objects. Personal data classes are classes that define instance variables that hold personal data and where one or more of these instance variables can be used to identify the data subject of the contained personal data. Personal data objects are instances of personal data classes.

Metadata must be provided for all personal data classes in the application. The metadata serves two purposes, it defines: (i) which types of personal data that the get and set methods of the personal data classes return and modify; (ii) how the data subject of a personal data object can be determined during runtime.

```
<?xml version="1.0"?>
<application name="EPR" >

   <vocabulary name="epr-voc.xml" >

  <personaldataclasses>

     <class name="epr.model.Pasient" managed=”y”>
          <datasubject>
             <finderclass classname="epr.subfinder.Journal" />
          </datasubject>
          <property name="id" >
             <type name="PERSON_ID" />
          </property>
          <property name="firstName">
             <type name="PERSON_NAME"/>
          </property>
         . . .
     </class>
         . . .
  </persondataclasses>
</application>
```

Figure 2: Example Metadata mapping file

The metadata maps the application data to a vocabulary so that the policy written in this vocabulary can be interpreted and enforced in the context of the application.

The vocabulary defines the entities (i.e. words) of the language used to express privacy policies. It defines valid data types (e.g. first_name), purposes (e.g. diagnosis), and actions (e.g. read, write). The application developers are free to choose a suitable vocabulary, preferably a standardized vocabulary for the application domain, if available.

Privacy metadata is supplied through metadata descriptor files, one file for each application. These files can be edited directly or through the Management interface of the Carnival Server.

Figure 2 shows an excerpt of the metadata descriptor file for the *EPR* application. It identifies the vocabulary used and the personal data classes of the application. The *Patient* class contains (at least) two instance variables holding personal data. The *id* instance variable is of type *PERSON_ID* and the instance variable *firstName* is of type *PERSON_NAME*. In addition there is a reference to the Data Subject Finder plug-in, *epr.subfinder.Journal*, which is used to identify the data subject of an instance of the class.

The metadata descriptor is created during application development and it may be edited during application deployment. Among others, during deployment it is determined which personal data classes that should be managed. How applications using Carnival are deployed and configured is described further in section VI.

Privacy metadata can also be provided through code annotations or through annotations of UML-diagrams constructed during the design phase. From these annotations the metadata descriptor of the application can be automatically generated. Figure 3 shows Java 1.5 annotations corresponding to the metadata descriptor file in Figure 2.

```
@no.nr.privacy.annotations.DataSubjectHelper
 (“epr.subfinder.Journal")
public class Patient{

     @no.nr.privacy.annotations.PersonalDataType("PERSON_ID")
     private int id;

   @no.nr.privacy.annotations.PersonalDataType(“PERSON_NAME”)
     private String firstName;
                .
     public int getId() {
        return id;
     }
     public String getFirstName() {
        return firstName;
     }

   }
```

Figure 3: Example annotated class

D. Extraction of privacy metadata during runtime

When the Privacy Manager evaluates an access request to personal data contained in a managed object the metadata of the requested data is retrieved. The types of the data requested is a static property, whereas the identity of the data subject is a dynamic property that can only be determined at runtime.

When a managed object is accessed the data types and its Data Subject Finder plug-in are looked up in the metadata descriptor file. Finally, the personal data object is passed to the Data Subject Finder plug-in that returns a string that identifies the data subject.

E. Purpose selection

The Purpose Selection service implements Carnival’s purpose selection logic. The Purpose Selection service’s behavior is defined by purpose rules. In the current implementation a user’s current purpose is determined as a

function of the user's roles and the method invoked by the user. The Purpose Selection service collects this information before method invocations.

The application should provide methods that are called when the user moves from one purpose to another. One way of accomplishing this is to design the application so that each task in the application is naturally delimited from the other tasks, for example through providing different GUI views for each task.

The user and application must of course have the same understanding of what the current purpose is. One way to achieve this is to have the application clearly display the current purpose and require that the user actively change this purpose if he/she disagrees. Carnival requires that applications provide Carnival with some method of communicating directly with users. More precisely, Carnival requires that applications provide callbacks, which Carnival uses to present the user's current purposes, and functionality for actively changing the current purpose.

VI. USAGE

The usage of Carnival can be divided into three phases: development, deployment, and operation.

A. Development

When developing an application using Carnival some design guidelines should be followed.

The application must take into consideration the fact that access to a method can be denied leading to an exception being thrown. Likewise, when an access has lead to obligations this is communicated to the application through an exception. This exception contains information about the executed obligation and the result of the access, which may have been affected by the obligation. For example, an approximation may be returned instead of the exact value. The information contained in exceptions allows the application to communicate to the user why access was denied and/or which obligations that have been executed.

Additionally, as stated before, the application should be designed in such a way that it is easy to capture the current purpose of users. Carnival also requires that Data Subject Finder plug-ins and GUI callbacks for purpose management are developed. Developers may also supply Condition Evaluators and Obligation Executors relevant for the application domain.

Furthermore, the privacy metadata descriptor file should be written, listing all personal data classes of the application, as described in section V.C.

B. Deployment

During application deployment, a vocabulary must be constructed or selected if one does not exist (see section V.C). The vocabulary used by the application may be adopted, a new may be constructed, or a standard vocabulary may be adopted. If the vocabulary bundled with the application is not used, the application's metadata descriptor file needs to be updated so that it is compliant with the new vocabulary.

Additionally, Carnival must be provided with a local privacy policy conforming to the chosen vocabulary, if not already available. For all obligation types included in the local policy, corresponding Obligation Executor plug-ins should be provided, and for each type of condition in the policy, a corresponding Condition Evaluator should be provided.

Finally, two application specific steps should be followed. Firstly, it should be decided which personal data objects that should be managed from the ones listed in the privacy metadata descriptor. Secondly, purpose rules should be provided (see section V.E).

C. Operation

When a relationship is established with a data subject (e.g. customer) his or her privacy preferences may be taken into account by creating an individual privacy policy for the data subject. This policy is added to the Carnival's repository of policies to be enforced.

VII. CONCLUDING REMARKS

This paper has presented Carnival, a framework that provides privacy protection functionality to applications. Carnival enables the implementation of privacy access control, which is different from the other types of access control, as discussed in section III. In addition, it provides functionality for producing audit trails to enable detection of privacy violations. Finally, it defines a number of services and plug-ins to support the enforcement of privacy policies: The Purpose and Purpose Selection services which provide information needed to evaluate policy rules, the Rule Engine and the Condition Evaluator which evaluate access requests to personal data, the Obligation Router and Executors which enforce obligations resulting from data access, and the Logger and Log services which handle audit trails.

However, note that Carnival does not include all functionality needed to enforce privacy policies. Organizations also need, among others, to provide channels for data subjects to access their personal data and data about its usage (Individual Participation Principle, see [12]), and measures to continually uphold the accuracy and completeness of the personal data stored (Data Quality Principle, see [12]).

Carnival fulfils the requirements listed in section IV, except support for pseudonyms, which has not been implemented yet. That is, it enables the retrieval of all information needed to evaluate privacy policy rules and determine whether or not requested access should be granted. Further, the access control logic is replaceable, and hence supports the implementation of different access control models and the use of different policy languages. In addition, services are defined to support different types of obligations.

The determination of a user's current purpose is handled through the inclusion of the Purpose Selection service. However, the design of this service needs to be further explored, as the determination of the current purpose is an intricate problem. We are not convinced that we have seen the

best solutions to this problem yet. One possibility we will investigate further is the use of a formal workflow-control system based on the use of Petri nets (see e.g. [9]). An advantage of this type of solution is that purposes can easily be defined across different applications.

Going forward, we plan to add support in Carnival for creation and management of pseudo domains, as proposed in [8]. Carnival will thus include functionality for generating pseudonyms, and regulating linkage between pseudonyms and the disclosure of the identities behind pseudonyms. This functionality is motivated by the fact that some functions in an organization may not need to have knowledge of information that directly identifies data subjects, typically the data subjects' name or identity number. Their work can equally well be carried out when data subjects are identified by pseudonyms. Additionally, different functions can be provided with different pseudonyms for the same data subject, preventing unauthorized linking and matching of information.

We also plan to further evaluate the usefulness and performance of Carnival. Questions related to how intuitively it is to integrate with applications, how well it scales, and how it affects performance, will be further examined.

VIII. References

[1] R. Agrawal, J. Kiernana, S. Ramakrishnan, and Y. Xu, Hippocratic Databases, IBM Almaden Research Center. Available at: http://www.almaden.ibm.com/software/dm/Hippocratic_Databases/hippocratic.pdf

[2] R. R. Arnesen and J. Danielsson: "A Framework for Enforcement of Privacy Policies," in Proceedings of the Nordic Security Workshop NORDSEC 2003, October 2003. Available at: http://publications.nr.no/A_Framework_for_Enforcement_of_Privacy_Policies.pdf

[3] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter (ed.), Enterprise Privacy Authorisation Language (EPAL 1.1), IBM, 2003. Available via http://www.zurich.ibm.com/security/enterprise-privacy/epal/

[4] D.F.C. Brewer and M.J. Nash, "The Chinese Wall Security Policy," IEEE Symposium on Security and Privacy, pp. 215-228, 1989

[5] Declarative Privacy Monitoring, IBM alphaWorks, http://www.alphaworks.ibm.com/tech/dpm

[6] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. Official Journal L 281, 23/11/1995, pp. 31-50. Available from http://europa.eu.int/eur-lex/en/index.html

[7] S. Fischer-Hübner and A. Ott., "From a Formal Privacy Policy Model to its Implementation," National Information Systems Security Conference (NISSC 98), 1998. Available at http://www.rsbac.org/niss98.htm

[8] R. Hes and J. Borking, (eds.), Privacy-enhancing technologies: The path to anonymity, Revised edition. ISBN: 90-74087-12-4. Registratiekamer, The Hague, August 2000

[9] K. Jensen, "Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use, Volume 1, Basic Concepts." Monographs in Theoretical Computer Science, Springer-Verlag, 1997

[10] G. Karjoth and M. Schunter, A Privacy Policy Model for Enterprises, 15th IEEE Computer Security Foundations Workshop, June 2002

[11] M. Hansen and A. Pfitzmann, Anonymity, Unobservability and Pseudonymity – A Proposal for Terminology, v0.21, Available at http://dud.inf.tu-dresden.de/Literatur_V1.shtml

[12] OECD, Guidelines on the Protection of Privacy and Transborder Flows of Personal Data, Available at http://www1.oecd.org/publications/e-book/9302011E.PDF

Measuring Anonymity Revisited

Gergely Tóth, Zoltán Hornák and Ferenc Vajda
Budapest University of Technology and Economics
Department of Measurement and Information Systems
H-1111 Budapest, XI., Magyar tudósok krt. 2.
Email: {tgm,hornak,vajda}@mit.bme.hu

***Abstract*— Anonymous message transmission systems are the building blocks of several high-level anonymity services (e.g. e-payment, e-voting). Therefore, it is essential to give a theoretically based but also practically usable objective numerical measure for the provided level of anonymity. In this paper two entropy-based anonymity measures will be analyzed and some shortcomings of these methods will be highlighted. Finally, source- and destination-hiding properties will be introduced for so called *local* anonymity, an aspect reflecting the point of view of the users.**

***Index Terms*— anonymity measure, local anonymity**

I. Introduction

Anonymous message sending techniques define a rapidlyn evolving area in privacy research. Such methods are required for many applications ranging from simple untracable e-mail communication to anonymous electronic voting and payment systems. The goal is to transport messages from senders to recipients, so that an attacker (ranging from simple observers to traffic shaping adversaries) can only guess the user-message relations with a small probability.

The aim of this paper is to draw attention to so called *local* anonymity. Recent papers have proposed entropy as a means of measuring the performance of different systems. Although *global* anonymity (i.e. how many potential candidates the adversary has to consider and what is their general distribution) can be quantified this way, the user's point of view is somewhat different: "I am only interested in my own messages and they should not be linked to me under any circumstances with a probability greater than a given treshhold". In response to this we should rather focus on the worst case scenario for a given message.

Another key issue is the aspect of the *user-defined treshhold*. This is a calibration metric, like Quality-of-Service that a system should satisfy when providing anonymity services. The aim in this paper is to clearly highlight the problem – the difference between local and global anonymity – and give some example solutions.

Although one can find several systems in the literature ([1], [2], [3]), each year newer and newer solutions are published ([4], [5]). In order to objectively compare them, appropriate theoretical measures are required. Another important requirement of the practical usability of such measures is that the measure should be easy to understand, not only by experts but also by users. The goal of the authors is to introduce source-hiding property for measuring sender anonymity and destination-hiding property for recipient anonymity and to compare them with existing measures.

A. Local vs. Global Anonymity

Serjantov & Danezis [6] and Díaz *et al* [7] proposed two similar information theory-based anonymity measures. By using the entropy of the attacker's probability distribution, they quantified how many bits of information an adversary needs in order to perfectly match a message to the respective user. This approach (later referred to as as *global* measure) aims to quantify the effort that is needed to totally compromize messages. (In the worst case missing bits of information can be substituted with brute force, where the required number of steps is the power of two.)

On the other hand, in this paper we argue that another approach – using the maximal probability as a measure – focuses better on the *local* aspect of anonymity. From the users' point of view this is more important, because they are interested only in their own messages and the probability of being compromized.

B. Outline of the Paper

In Section II. we briefly introduce previous work in the field of anonymity measures and then analyze the shortcomings of these approaches in Section III. In Section IV. the proposed source- and destination-hiding properties will be introduced. We will show that they represent worst-case anonymity measures, mainly focusing on the local aspect of the user's view. Finally the analysis of a continuous time system (the PROB-channel) closes the paper with the calculations for the different anonymity metrics, which have been introduced.

II. Background

This section gives a short introduction to the background of measuring anonymity. First let the informal summary of an anonymous message transmission system follow. This will define the terms and expressions we are going to use in this paper. Later in this section different previously published entropy-based anonymity measures will be described.

A. Anonymous Message-sending Scenario

In an anonymous message-sending scenario we have the following setting: *senders* send *messages* to *recipients* using the intermediate *anonymous message transmission system*. This anonymous message transmission system cyptographically *transforms*, *delays* and *mixes* the messages sent by the

senders according to the implemented algorithm and eventually delivers them to the recipients.

On the other hand there is an *adversary*, who may see messages sent by the senders and also those delivered to the recipients. His aim is to match the delivered ones to the senders (accoring to [8] in this case sender anonymity is compromized) or the sent messages to the recipients (recipient anonymity).

In order to render the effors of the adversary more difficult, the parties use diffent encryption algorithms, uniformly sized messages and dummy traffic [9].

Considering the adversary different attacker models can be taken into account: mighty ones may perceive the whole network at all times, whereas a less pessimistic approach may consider attackers with limited access to a fraction of the whole network. Another important aspect is whether the adversary is active (i.e. may delay, create, delete or alter messages) or only passive (i.e. can only eavesdrop). When calculating the level of anonymity provided by a system it is an important aspect to note against what kind of adversary the metrics hold.

Furthermore we assume that the adversary performs a probabilistic attack: he computes probabilities that indicate, to what extent messages correspond to senders or recipients according to his knowledge. Finally the adversary marks the most probable sender/recipient as his guessed user for a certain message.

B. Anonymity Measures

Based on the model of an anonymous message transmission system the definition of anonymity was given by Pfitzmann and Köhntopp [8]:

> Anonymity is the state of being not identifiable within a set of subjects, the *anonymity set*.
> [...]
> Anonymity may be defined as the unlinkability of an IOI[1] and an identifier of a subject.

The first publications aiming to quantify the level of anonymity provided by the described systems used the size of the anonymity set as the measure (e.g. [3]). Since the probabilities might not be uniformly distributed, the size of the set does not perfectly reflect the achieved anonymity as it was pointed out with the practical example of the pool mix in [10].

Based on the above observation Serjantov & Danezis introduced entropy for measuring anonymity [6]. They used the following model:

> *Definition 1:* Given a model of the attacker and a finite set of all users Ψ, let $r \in \mathcal{R}$ be a role for a user ($\mathcal{R}$=$\{$sender, recipient$\}$) with respect to a message $\mathcal{M}$. Let $\mathcal{U}$ bet the attacker's a-posteriori probability of users $u \in \Psi$ having the role r with respect to $\mathcal{M}$.

With this in mind the measure for both sender and recipient anonymity was defined as follows:

[1]Item Of Interest, i.e. a message

> *Definition 2:* The effective size S of an r anonymity probability distribution $\mathcal{U}$ is equal to the entropy of the distribution. In other words
>
> $$S = -\sum_{u \in \Psi} p_u \log_2 p_u \qquad (1)$$
>
> where $p_u = \mathcal{U}(u, r)$.

In the rest of the paper this anonymity measure will be referred to as the *simple entropy* measure.

Díaz et al. followed a slightly different (extended) approach [7], whereas they only considered sender anonymity. Let $\mathcal{A}$ represent the anonymity set of a certain message $\mathcal{M}$, i.e. $\mathcal{A} = \{u | (u \in \Psi) \wedge (p_u > 0)\}$. Furthermore let N be the size of the anonymity set, i.e. $N = |\mathcal{A}|$. Their defintion was the following:

> *Definition 3:* The degree of anonymity provided by a system is defined by
>
> $$d = \frac{H(X)}{H_M} \qquad (2)$$
>
> For the particular case of one user we assume d to be zero.

With the symbols defined above $H(X) = S$ and $H_M = \log_2 N$. We will refer to this measure as the *normalized entropy* measure.

In both cases 0 means absolutely no anonymity (i.e. the attacker knows with 100% the sender of a message). In the simple entropy case maximal anonymity is achieved when $S = \log_2 N$ and with normalized entropy when $d = 1$.

III. Shortcomings of Existing Anonymity Measures

In the following the previously introduced entropy based measures will be evaluated and some shortcomings will be pointed out:

- For both measures two probability distributions will be given that have the same level of anonymity according to the respective measure, but practically provide very different anonymity considering the local aspect, i.e. the worst case for one particular user.
- It will be shown that non-desirable systems can approach optimal systems according to the entropy based measures.

A. Simple Entropy

Recall that accoring to the measure of simple entropy the level of anonymity is given by S, see (1). First, two distributions will be shown that have the same entropy but behave remarkably differently considering the provided anonymity from one user's point of view.

Now let's define the following two anonymity systems:

1) In the first system the probability distribution (D_1) is uniform among m users, i.e. $D_1 \colon p_u = \frac{1}{m}$.
2) In the second system we have a different distribution (D_2) among n users: for the sake of the example for the

TABLE I
CORRESPONDING n-m VALUES YIELDING THE SAME AMOUNT OF SIMPLE ENTROPY

m	n	S
10	26	3.3219
20	101	4.3219
50	626	5.6439
100	2501	6.6439

actual sender the probability is 50% and the others are uniformly distributed [2]. This yields the following:

$$D_2 \colon p_u = \begin{cases} 0.5 & \text{for the actual sender,} \\ \frac{0.5}{n-1} & \text{otherwise.} \end{cases}$$

Now let's choose m and n so that the resulting entropy is the same. For this we have to solve $S^m_{D_1} = S^n_{D_2}$, which is expanded in the following equation:

$$-\left[m\frac{1}{m}\log_2\frac{1}{m}\right] = -\left[(n-1)\frac{0.5}{n-1}\log_2\frac{0.5}{n-1} + 0.5\log_2 0.5\right] \quad (3)$$

The result can be seen in (4):

$$n = \frac{m^2}{4} + 1 \quad (4)$$

Some example numerical values are shown in Table I. In order to visualize the problem, let's have a look at the example with $m = 20$. According to the definitions in such a system with uniformly distributed probabilities (D_1) the attacker has 5% (i.e. $p_u = \frac{1}{20} = 0.05$) chance to guess the sender of a message. This system provides anonymity with $S = 4.3219$ bits.

On the other hand, let's have a look at the second system (D_2). Here for each delivered message the attacker knows that a particular sender sent the message with 50% certainty and another 100 senders could have sent it with 0.5%.

The two systems clearly perform differently considering the local aspect of anonymity, but have the same value with simple entropy. With D_1 distribution statistically seen on the long term an attacker can guess the sender of a message every 20^{th} time correctly, whereas with D_2 distribution he is going to successfully guess the sender of every second message.

The second point to show is that non-desirable systems can achieve an arbitrarily high entropy. From (1) and (4) it is clear that for an arbitrary value of S a corresponding D_2 distribution can be constructed, where $n = 4^{S-1} + 1$.

Summarized, the main problem with this entropy based measure is that it tries to quantify the amount of information that is required to break *totally* the anonymity of a message, i.e. to definitely identify actual sender of a message. However in practice we have to consider an attacker successful, if he can guess the sender of some selected messages with a good probability, in specific cases significantly greater than in the case of the uniform distribution (i.e. $p_u \gg \frac{1}{N}$).

[2]The concrete probability of 0.5 was chosen in order to simplify the resulting equations.

B. Normalized Entropy

To demonstrate similar shortcomings of the normalized entropy measure first we show two systems with the same value of d, however with remarkably different local anonymity. Due to the normalization we have to notice that following from the definition of d in order to obtain the same results for the two constructions the quotient of the entropy and the the logarithm of the anonymity set size should remain the same. This can be achieved in the easiest way by having the same entropy as well as the same anonymity set size.

For demonstration purposes let's consider the following two systems:

1) In the first system we have the distribution (D_2) known from the previous example: n users are involved; for the actual sender the probability is 50% and the others are uniformly distributed. This yields the following distribution:

$$D_2 \colon p_u = \begin{cases} 0.5 & \text{for the actual sender,} \\ \frac{0.5}{n-1} & \text{otherwise.} \end{cases}$$

2) In the second case we have a new distribution (D_3): there are n users as well, x of them having a probability P_A and $n - x$ of them with probability P_B. The characteristic parameters for such a distribution are x and P_S, being the sum of the P_A probabilities of the x users. The following distribution is given this way:

$$D_3 \colon p_u = \begin{cases} P_A = \frac{P_S}{x} & \text{for the } x \text{ users,} \\ P_B = \frac{1-P_S}{n-x} & \text{otherwise.} \end{cases}$$

This second distribution can be explained as follows: with P_S probability the sender of the message is the member of a sub-anonymity-set $\mathcal{A}_A$ with x members and uniformly distributed probabilities P_A. On the other hand with $1-P_S$ probability the sender of the message is the member of the other sub-anonymity-set $\mathcal{A}_B$ with $n-x$ members and also uniformly distributed probabilities P_B.

In order to find suitable distributions the equation below has to be solved (notice that for the distribution D_2 the entropy was caluclated in (3)):

$$\frac{-\left[(n-1)\frac{0.5}{n-1}\log_2\frac{0.5}{n-1} + 0.5\log_2 0.5\right]}{\log_2 n} = \frac{-\left[xP_A\log_2 P_A + (n-x)P_B\log_2 P_B\right]}{-log_2 n} \quad (5)$$

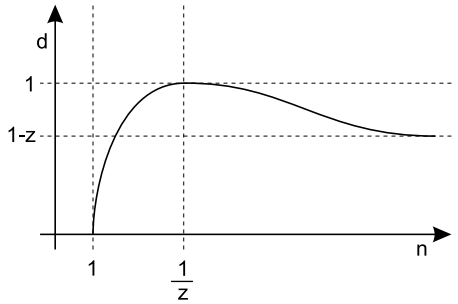
It is clear that for this scenario we have three variables: n, x and P_S. For a concrete example, x was chosed to be $x = \frac{m}{2}$, where $m = \sqrt{4n-4}$ (see (4)) and the respective P_S was calculated (see Table II).

To imagine the two systems, let's look at the case $m = 20$. With this we get an anonymity set $\mathcal{A}$ with $n = 101$ users. For

TABLE II
CORRESPONDING n-x-P_S VALUES YIELDING THE SAME NORMALIZED ENTROPY

m	n	x	P_S	P_A	P_B	d
10	25	5	0.832213	0.166442	0.007989	0.706727
20	101	10	0.865184	0.086518	0.001481	0.649112
50	626	25	0.890594	0.035623	0.000182	0.607518
100	2501	50	0.903463	0.018069	0.000039	0.588561

Fig. 1. d as a function of n with distribution D_2'



both systems the normalized entropy gives $d = 0.649112$ as a measure for the anonymity.

In case of the first system (D_2) for the actual sender of the message $p_u = 0.5$, thus the attacker knows with 50% certainty of the sender, for the other 50% he has 100 possible users with 0.5% certainty uniformly distributed.

On the other hand for the second system (D_3) we have two sub-anonymity-sets. For $\mathcal{A}_A$ we have 10 users with probabilities P_A of roughly 8.7%, yielding together P_S of 87%. Furthermore we have the other sub-anonymity-set $\mathcal{A}_B$, consisting of 91 users with an overall probability of about 13% uniformly distributed in quantities of 0.15% as P_B.

Another important point is to show that non-desirable systems exist in arbitrarily small vicinity of the optimal $d = 1$. For this let's consider a slightly modified version of the D_2 distribution that we will refer to as D_2'. In this distribution n users are involved; for the actual sender the probability is z and the others are uniformly distributed. This yields the following:

$$D_2' \colon p_u = \begin{cases} z & \text{for the actual sender,} \\ \frac{1-z}{n-1} & \text{otherwise.} \end{cases}$$

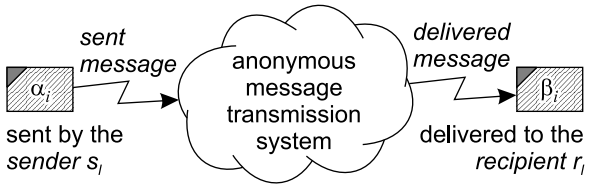
The degree of anonymity provided according to the normalized entropy is as follows:

$$d = \frac{-\left(z \log_2 z + (n-1)\frac{1-z}{n-1} \log_2 \frac{1-z}{n-1}\right)}{\log_2 n} \tag{6}$$

After analyzing d as a function of n (as seen on Fig. 1) we can determine the following:

- With one user $d = 0$ is trivial.
- With $\frac{1}{z}$ users $d = 1$ is maximal. This is evident as in this case we have uniform distribution.
- Finally it can be proven that $\lim_{n\to\infty} d = 1 - z$ and that on the interval $\left[\frac{1}{z}, \infty\right)$ $d > 1 - z$.

Fig. 2. Message sending with the anonymous message transmission system



With the above in mind we can see that even with a system, where $n \gg \frac{1}{z}$ the degree of anonymity is above the treshhold, i.e. $d > 1 - z$, thus systems can get arbitrarily close to the optimal $d = 1$ and yet they are non-desirable in the sense that there are users whose level of local anonymity is above an acceptable probability.

IV. LOCAL ANONYMITY MEASURE

In the previous section shortcomings of the information theory based *global* anonymity metrics were evaluated. In those cases it was quantified, how much additional information an attacker needs in order to definitely identify the user corresponding to the message (i.e. its sender or recipient).

On the contrary our argument is that an attacker is already successful if he can guess these links with a good probability (that is over a certain acceptable treshhold).

Before defining local anonymity measures, the used terms will be introduced. In the analyzed system *senders* ($s_l \in S$) transmit encrypted *sent messages* ($_j \in \varepsilon_S$) to the anonymous transmission system. After transforming (re-encoding) and delaying them the *delivered messages* ($\beta_k \in \varepsilon_R$) reach the *recipients* (see Fig. 2.). Time of sending is indicated by $t_S(\ _j)$, similarly time of receipt is $t_R(\beta_k)$. Sender of a sent message is denoted by $S(\ _j)$ and the recipient by $R(\beta_k)$.

The adversary has two aims: to break sender anonymity by computing the probabilities P_{β_k,s_l} (i.e. what is the probability that β_k was sent by s_l) and to break recipient anonymity by computing $P_{\ _j,r_l}$ (i.e. r_l received $\ _j$).

For this scenario in [5] the destination- and source-hiding properties were defined for sender and recipient *local* anonymity.

Definition 4: A system is *source-hiding* with parameter Θ if the adversary cannot assign a sender to a delivered message with a probability greater than Θ, i.e. if

$$\forall_{\beta_k} \forall_{s_l} \left(P_{\beta_k,s_l} \leq \Theta\right) \tag{7}$$

holds.

Definition 5: A system is *destination-hiding* with parameter Ω if the adversary cannot assign a recipient to a sent message with a probability greater than Ω, i.e. if

$$\forall\ _j \forall_{r_l}\ \ P_{\ _j,r_l} \leq \Omega \tag{8}$$

holds.

It is important to note that one cannot draw grounded conclusions about the local anonymity from global anonymity measures as it was shown in the previous section (i.e. for

arbitrarily high global anonymity systems with non-desirable local anonymity exist, where in the worst case the identity of some users can be guessed with an unacceptably big probability).

On the contrary we will show that from the local anonymity measures we can draw conclusions for the global anonymity meausres as well. In the following we will deduce results for the sender anonymity from the source-hiding property but since it is symmetric for the destination-hiding property, similar equations can be stated as well for the recipient anonymity.

Theorem 1: For a system with source hiding property with parameter Θ the inequality below holds:

$$S \geq -\log_2 \Theta \tag{9}$$

Informally this theorem means that a system of source-hiding property with parameter Θ is in the global sense at least as strong as a system with $\frac{1}{\Theta}$ users and uniformly distrbuted probabilities.

Proof: First from the definition (7) it follows that $\forall_{u\in\Psi}(0 < p_u \leq \Theta \leq 1)$. Therefore since the logarithm function is monotonic in the interval $(0,\infty) \Rightarrow \forall_{u\in\Psi}(\log_2 p_u \leq \log_2 \Theta) \Rightarrow \forall_{u\in\Psi}(-\log_2 p_u \geq -\log_2 \Theta)$.

With this (1) can be rewritten:

$$\begin{aligned} S &= -\sum_{u\in\Psi} p_u \log_2 p_u \\ &\geq -\sum_{u\in\Psi} p_u \log_2 \Theta \\ &= -\log_2 \Theta \sum_{u\in\Psi} p_u \\ &= -\log_2 \Theta. \end{aligned}$$

■

With the combination of (2) and (9) a lower limit to d can be given as well:

$$d \geq -\log_N \Theta \tag{10}$$

as $d = \frac{S}{\log_2 N} \geq -\frac{\log_2 \Theta}{\log_2 N} = -\log_N \Theta$.

V. Analysis of the PROB-channel

The PROB-channel is an example for an anonymous message transmission system introduced in [5]. In order to show how the introduced anonymity metrics work with practical anonymity systems in this section the PROB-channel will be introduced and its anonymioty level will be analyzed.

A. Brief Defintion of the PROB-channel

The PROB-channel is a continuous time system, where messages are processed independently. Once a message enters the channel, a delay will be calculated for it and after that time has passed the message leaves. This delay δ in the system is a probability variable with a given density function $f(\delta)$ (i.e. $\int_0^\infty f(\delta)d\delta = 1$). In order to guarantee real-time probabilities, the delay in the PROB-channel has a pre-defined maximum ($\delta_{\max}$). On the other hand considering real systems a minimal delay ($\delta_{\min}$) was also defined:

$$\forall_{\delta\notin(\delta_{\min},\delta_{\max})} f(\delta) = 0 \tag{11}$$

In order to simplify further equations first two sets need to be defined. With μ_{β_k} the set of sent messages is meant that might have left the channel as β_k (12), whereas η_{β_k,s_l} denotes the subset of μ_{β_k}, which was sent by the specific sender s_l (13).

$$\mu_{\beta_k} = \{\ _j | (t_R(\beta_k) - \delta_{\max}) < t_S(\ _j) < (t_R(\beta_k) - \delta_{\min})\} \tag{12}$$

$$\eta_{\beta_k,s_l} = \{\ _j | (\ _j \in \mu_{\beta_k}) \wedge (S(\ _j) = s_l)\} \tag{13}$$

B. The Attacker Model – A Passive Observer

As an attacker model let's consider a passive observer: he can eavesdrop on all connections but does not alter, delete or delay messgaes.

Aim of the passive observer is to link delivered messages to the senders by computing the probabilities P_{β_k,s_l}. The most effective solution is summarized in (14):

$$P_{\beta_k,s_l} = \frac{\sum_{\ _j\in\eta_{\beta_k,s_l}} f\left(t_R(\beta_k) - t_S(\ _j)\right)}{\sum_{\ _j\in\mu_{\beta_k}} f\left(t_R(\beta_k) - t_S(\ _j)\right)} \tag{14}$$

Of course the attacker chooses s_i as the sender for β_k where $s_i = \max_{s_l} P_{\beta_k,s_l}$.

C. Methods to Ensure Local Anonymity

It it clear from (14) that in the general case no hard guarantee can be given about P_{β_k,s_l}. The main problem comes from the real-time requirement: even if only one message is in the channel, it has to be delivered before the maximal delay expires. Thus in unfortunate cases the adversary has an easy task.

In order to ensure that there are enough messages to form a sufficiently large anonymity set for each message the only solution is to enforce continuous message sending. The MIX/MAX property was defined for this purpose.

Definition 6: A system fulfills the criteria of the MIX/MAX property with parameters $\tau_{\min}$ and $\tau_{\max}$ ($\tau_{\min} < \tau_{\max} < \delta_{\max}$) if all senders send at least one message in every $\tau_{\max}$ interval but no sender sends more than one message in any $\tau_{\min}$ interval.

With the above definiton the amount of messages can be fine-tuned and also the fraction, a specific user reaches from the whole amount of messages, can be set. It was shown in [5] that with the MIX/MAX property local anonymity can be ensured, see (15) for the source-hiding property.

$$\Theta = \frac{\sum_{i=1}^{\Delta_{\min}} \max_{(i-1)\cdot\tau_{\min}\leq q\leq i\cdot\tau_{\min}} f(q)}{N \cdot \sum_{i=1}^{\Delta_{\max}} \min_{(i-1)\cdot\tau_{\max}\leq q\leq i\cdot\tau_{\max}} f(q)} \tag{15}$$

where $\Delta_{\max} = \lfloor \frac{\delta_{\max}-\delta_{\min}}{\tau_{\max}} \rfloor$ and $\Delta_{\min} = \lceil \frac{\delta_{\max}-\delta_{\min}}{\tau_{\min}} \rceil$.

D. The Optimal System

Sticking to the real-time guarantee if was proven that the optimal delay characteristics is achieved if the channel uses the uniform density function (16).

$$f(\delta) = \frac{1}{\delta_{\max} - \delta_{\min}} \tag{16}$$

It is interesting to note that by changing the guaranteed maximal delay to a softer mean delay of a and enabling even infinite delays (of course with small probability) another density function proves to be the optimal, namely the exponential one (17) as first proposed by Kesdogan et al. for the SG-MIX [11] and then proven to be optimal by Danezis [12].

$$f(\delta) = \frac{1}{a} e^{-\frac{1}{a}\delta} \tag{17}$$

E. Quantified Anonymity of the PROB-channel

In the optimal case of uniform delay density (16) and MIX/MAX property (15) the local anonymity can be guaranteed efficiently. If $N \geq \frac{\tau_{\max}}{\tau_{\min}}$ then the following equation gives a good approximation:

$$\Theta \approx \frac{\tau_{\max}}{N \cdot \tau_{\min}} \tag{18}$$

Using results (9) and (10) from the previous section the following guarantees can be given for the global anonymity:

$$\begin{aligned} S &\geq -\log_2 \Theta \\ &= \log_2 N - \log_2 \frac{\tau_{\max}}{\tau_{\min}} \end{aligned} \tag{19}$$

$$\begin{aligned} d &\geq -\log_N \Theta \\ &= 1 - \log_N \frac{\tau_{\max}}{\tau_{\min}} \end{aligned} \tag{20}$$

From these results it is clear that the main calibration possibility of the PROB-channel is the fraction $\frac{\tau_{\max}}{\tau_{\min}}$. It is obvious that if $\tau_{\max} = \tau_{\min}$ then an overall optimum can be reached where the anonymity set of each message is maximal and the probabilities P_{β_k, s_l} are uniformly distributed among all possible senders:

$$P_{\beta_k, s_l} = \frac{1}{N} \tag{21}$$

VI. Conclusion

The main focus of this paper was to introduce the term *local* anonymity and appropriate metrics for measuring it: the source- and destination-hiding properties. Previous information theory based anonymity measures aimed to quantify the number of bits required by an adversary to completely trace back a message. On the contrary we argue that an attacker is already successful if he can compromize messages with a probability above a certain treshhold for some of the users – which from the local aspect of the users is unacceptable, however possible in unfortunate cases of entropy-based global anonymity measures.

With this paper the importance of the local aspect was underlined and via a practical example of the PROB-channel enlightend. Future work should be carried out in order to analyze other practical solutions as well from this local point of view.

References

[1] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 4, no. 2, pp. 84–88, February 1981.

[2] M. Reiter and A. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, June 1998.

[3] O. Berthold, H. Federrath, and S. Köpsell, "Web mixes: A system for anonymous and unobservable internet access," in *Designing Privacy Enhancing Technologies*, ser. Springer-Verlag, LNCS, H. Federrath, Ed., vol. 2009, Berkeley, CA, 2001, pp. 115–129.

[4] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[5] G. Tóth and Z. Hornák, "Measuring anonymity in a non-adaptive, real-time system," in *Proceedings of Privacy Enhancing Technologies (PET2004)*, ser. Springer-Verlag, LNCS, Forthcoming, 2004.

[6] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Proceedings of Privacy Enhancing Technologies (PET2002)*, ser. Springer-Verlag, LNCS, P. Syverson and R. Dingledine, Eds., vol. 2482, San Francisco, CA, April 2002.

[7] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Proceedings of Privacy Enhancing Technologies (PET2002)*, ser. Springer-Verlag, LNCS, P. Syverson and R. Dingledine, Eds., vol. 2482, San Francisco, CA, April 2002, pp. 54–68.

[8] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity – a proposal for terminology," in *Designing Privacy Enhancing Technologies*, ser. Springer-Verlag, LNCS, H. Federrath, Ed., vol. 2009, Berkeley, CA, 2001, pp. 1–9.

[9] O. Berthold and H. Langos, "Dummy traffic against long term intersection attacks," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*, R. Dingledine and P. Syverson, Eds., vol. 2482. San Francisco, CA: Springer-Verlag, LNCS, April 2002.

[10] A. Serjantov and R. E. Newman, "On the anonymity of timed pool mixes," in *Security and Privacy in the Age of Uncertainty*. Athens, Greece: Kluwer, May 2003, pp. 427–434, (Workshop on Privacy and Anonymity Issues in Networked and Distributed Systems).

[11] D. Kesdogan, J. Egner, and R. Büschkes, "Stop-and-go MIXes: Providing probabilistic anonymity in an open system," in *Proceedings of Information Hiding Workshop (IH 1998)*, ser. Springer-Verlag, LNCS, vol. 1525, Berkeley, CA, 1998.

[12] G. Danezis, "The traffic analysis of continuous-time mixes," in *Proceedings of Privacy Enhancing Technologies (PET2004)*, ser. Springer-Verlag, LNCS, Forthcoming, 2004.

The Location Information Preference Authority: Supporting user privacy in location-based services

Anand S. Gajparia, Chris J. Mitchell and Chan Yeob Yeun

***Abstract*— To offer location-based services, service providers need to have access to Location Information (LI) regarding the users which they wish to serve; this is a potential privacy threat. Constraints, i.e. statements limiting the use and distribution of LI, that are securely bound to the LI, have been proposed as a means to reduce this threat. However, constraints may themselves reveal information to any potential LI user — that is, the constraints themselves may also be a privacy threat. To address this problem we introduce the notion of a LI Preference Authority (LIPA). A LIPA is a trusted party which can examine LI constraints and make decisions about LI distribution without revealing the constraints to the entity requesting the LI. This is achieved by encrypting both the LI and the constraints with a LIPA encryption key. This ensures that the LI is only revealed at the discretion of the LIPA.**

I. Introduction

As the potential for services provided by mobile phones advances [1], it may no longer be appropriate to call such devices mobile phones. Mobile phones already provide far more than the voice communications for which they were originally designed. Text messaging and video download are just two examples of the range of services which are now available to the consumer. We therefore use here the more general term ‘mobile device’.

Amongst the features currently available in mobile devices are location-based services. Location-based services may also be provided to devices which are not mobile, such as desktop PCs. We thus refer here to ‘user devices’, which include both mobile and non-mobile devices. We can then define a location-based service as a service based on the location of a user device [2]. In order to facilitate the provision of such a service, it is necessary that LI is made available to one or more entities; this is at the root of the privacy issues with location-based services.

To provide a location-based service, it may be necessary for LI regarding the user to be passed to an entity with whom the user has little or no basis for a trust relationship. It is unreasonable, however, for a user to be forced to allow its LI to be provided to any entity which requests it, since this would leave the end user with no control over its LI, which is, of course, personal information. It is also unreasonable for a service provider to freely distribute the LI of a user to other entities without permission.

This paper introduces a mechanism designed to enable the end user to take advantage of the convenience of location-based services, and yet also control the way LI is used, stored and distributed.

We begin by introducing constraints [3]. The use of constraints is a technique which allows a user to dictate the way in which LI is managed. We look at some of the disadvantages of constraints which motivate the design of the scheme proposed in this paper.

We next look at the security requirements for methods to enable control of, and privacy for, LI. With this in mind, the notion of a Location Information Preference Authority (LIPA) is introduced. A LIPA is essentially a trusted party which helps control the distribution of LI and accompanying constraints. LI is distributed to service providers in the form of an ‘LI token’. The LI token includes LI securely bound to its constraints. The LI and constraints are also encrypted using the LIPA’s public key, ensuring that unauthorised entities cannot see this information.

We then look at how the LIPA mechanism may be used to address problems with constraints, LI control, and privacy.

II. Previous work

In previous work, a variety of different aspects of security for location-based services have been considered. Existing schemes for LI privacy are in many cases geared towards the available wireless technology architectures. These include IEEE 802.11 [4] networks, mobile IP [5] and GSM networks [6].

Myles *et al.* [7] describe constraints which may be used to control the distribution of location information, although they do not describe cryptographic protection mechanisms to provide privacy. A user registers their privacy requirements with a location server, referred to as LocServ. Entities which require location information make requests to the LocServ, providing their own privacy policies. Based on this, the LocServ can then make a decision whether or not to provide location information. This mechanism does not provide any means for entities to pass on information to other entities.

Aura *et al.* [8] investigate authenticated location information in the Mobile IPv6 protocol. Aura *et al.* see authenticated location information as a defence mechanism against false routing information, which could lead to other forms of attack. The subject of authentic location information is also discussed in [9]. The discussion in this latter paper concerns the location of GSM devices. The motivation is to support location-based access control mechanisms and the inclusion of LI in audit logs. By contrast, the primary objective of this paper is the privacy of personal location information.

The Internet Engineering Task Force (IETF) geopriv working group is developing a general model for the protection of

location information [10]. This model is primarily concerned with securing the Location Object (LO), which encompasses location information and other necessary information which may include constraints. They describe a general model which addresses the security requirements for such an object, encompassing a variety of scenarios. Our LIPA model looks at a specific scenario for a generally distributed LI token containing constraints and LI.

III. LI, Constraints and their limitations

A. LI entities

Below are descriptions of the entities in our simple model of a system in which LI is used [3].

- **Location Information (LI).** This is data which provides information regarding an LI subject's location. LI may occur in many forms. In general, we can divide LI into two types, namely *Inferred* LI and *Actual* LI. Actual LI refers to a directly calculated geographical location. This type of data indicates, to some degree of accuracy, the physical location of an LI subject. Inferred LI is, by contrast, obtained by implication. For example, if a user is present on a network, this implies that they are likely to be within an certain vicinity, although no specific calculation of geographical LI has taken place.
- **LI subject.** An LI subject is the entity about whom location information is being gathered, managed and used. This entity is most commonly a human user.
- **Location-Based Service (LBS).** This is a service based on LI, e.g. a vehicular navigation service.
- **Location Information Preference Authority (LIPA).** This entity, discussed in more detail in Section IV, acts like a trusted party on behalf of the LI subject. There may exist many LIPA entities, where the LI subject will typically be able to choose its preferred LIPA. Where an LI subject device has the capability, this device could itself act as the LIPA.
- **Malicious Party.** This is an entity with malicious intent. A malicious party may act as a threat to the confidentiality, integrity or availability of LI for one or more LI subjects.
- **User Device (UD).** This entity is a device with which the LI subject may interact, e.g. to invoke a location-based service. Such a device may either be static, e.g. a desk top computer, or more typically mobile, such as a mobile phone or Personal Digital Assistant (PDA). It is, in fact, this device regarding which LI is generated rather than the user him/herself, since there is typically no way to directly measure the location of individuals. Thus this entity is a key part of the model.
- **LI gatherer.** This is an entity which gathers or possesses LI about an LI subject and then creates an LI token using this information. The LI token is discussed further in section IV.

  A GPS receiver is an example of part of an LI gatherer, as it obtains location data. An entity in a GSM network which keeps signalling data for a UD is also an example of part of a LI gatherer. Although a GSM network does not normally pass on this LI (except in certain special cases), it certainly possesses such information, and could, in an appropriate environment, be a valuable source of LI for commercial use. Other examples of methods used to generate LI can be found in [11].
- **Regulator/Legal authority.** This is an entity which exerts legal or regulatory control over the management and use of LI. This includes telecommunications regulators, data privacy authorities, law enforcement bodies, and auditors.

B. Privacy and LI

It is becoming increasingly difficult to keep personal information private [12]. It does not help that users have a variety of incentives to surrender it. Shoppers frequently use loyalty cards in exchange for a variety of benefits. Using these cards, information regarding times at which users shop, what they buy, and where they buy from, may be recorded [13]. In this case, shoppers typically have the option of denying access [14] to such information by simply not using these loyalty cards. However, once a customer decides to use a loyalty card, restricting access to any information gathered from it becomes difficult. This problem applies to all forms of personal information, including LI, and does not only apply to loyalty cards.

Almost certainly the main LI security issue is the potential breach of privacy arising from the transmission of LI to entities not trusted by the LI subject. It is important to note that a breach of user privacy only occurs when the relationship between the identity of the LI subject and the LI can be established. Anonymous communication, where a user may use a resource or service without disclosing its identity, or communication using a pseudonym, where a user may use a resource or service without disclosing its user identity but can still be accountable for that use, could overcome this problem. However, in many cases, e.g. for billing, it is difficult to use anonymous or pseudonymous communication. Moreover, whilst many proposals for protecting location privacy rely on anonymisation of the LI subject, this does not seem as if it will be a solution of general applicability – many, conceivably most, location-based services will require the service provider using LI to be able to associate the information with a particular LI subject. Thus we throughout assume that the (authorised) user of LI is permitted to learn the association between the LI and the LI subject.

Another privacy issue is analogous to the problem of 'spam', i.e. the receipt of unsolicited messages. This already poses a huge problem in email systems [15], and has also started to become an issue in other domains, e.g. mobile text messaging. This is a problem which may also migrate to location-based services and thereby become even more intrusive. For example, service providers wishing to advertise their services [16] may use LBSs to send unsolicited messages to LI subjects in a given area.

To resolve these issues, LI should only be provided to entities authorised by the LI subject.

C. Constraints

Constraints are simply statements, bound to LI, which may be used to help control the use, storage and distribution of this LI [3].

An LI subject may, for example, want to limit the period of time an entity stores their LI. This will prevent entities collating data to provide information about the LI subject's travel habits. Storage time may be limited either by stating in the constraints the amount of time that the LI may be kept from a specified start point, or by stating a point in time after which the LI must be deleted. In the first case, the start point may be indicated by including a time stamp in the constraints, e.g. the time at which the LI was generated. However, as previously discussed in [3], placing a time stamp in the constraints allows receiving entities to learn the time at which LI was generated, and so the time when the LI subject was at a particular location. By contrast, a mechanism stating the time when the LI expires will limit the information revealed, as the time at which the LI subject was at a location cannot be precisely determined.

Limiting the distribution of LI ensures that LI is only sent to entities authorised by the LI subject. Restrictions on LI distribution may be specified either by stating the entities who are authorised to receive the LI, or by listing the entities not authorised to receive the LI. However, statements about permitted distribution give a receiving entity knowledge about relationships between the LI subject and other entities. For example, it enables entities to know which other entities are trusted by the LI subject and those which are not.

LI use may be restricted by stating how LI is or is not to be used. For example, an LI subject may only want their LI used for navigation purposes, and the constraints could state this. Conversely, the constraints could contain a negative statement indicating that, for example, the LI is not to be used for advertising purposes. These types of statement also provide information about the preferences of an LI subject, i.e. they are themselves a potential breach of user privacy.

Thus, providing information about how LI is to be managed allows personal information to be divulged. This is because the preferences of an LI subject are themselves personal information. Thus, in order to fully protect user privacy, the statements in the constraints must somehow be enforced without divulging the contents of the constraints to the LI consumers.

IV. A MECHANISM TO PROVIDE SECURITY FOR CONSTRAINTS

In this section the LIPA-based mechanism, providing privacy control for LI and associated constraints, is described.

A. Overview of the mechanism

In order to ensure that the information held within the constraints remains private, we propose the use of a trusted party which we call a Location Information Preference Authority (LIPA). The LI gatherer is assumed to be in possession of the list of preferred LIPAs for each LI subject for which it generates LI. This is an indication of the LIPAs trusted by the LI subject. The LI gatherer must be trusted by the LI subject to act according to its wishes.

1) **LI gathering.** The first step in our mechanism involves the provision of LI by the gatherer. The LI gatherer may be at any location, including in the UD itself. The LI gatherer may obtain LI in response to a request by an LBS provider or an LI subject, or it may constantly collect LI for a large number of LI subjects.
2) **LI token generation.** The LI gatherer then creates what we refer to as an LI token. This includes both LI and accompanying constraints. The LI and constraints are encrypted by the LI gatherer using the public key of the LIPA. This ensures that only the LIPA is able to view this information. Also contained within the scope of the token is information which helps to identify both the LI subject and the LIPA, together with a unique token identifier. The LI token includes the signature of the LI gatherer, guaranteeing the integrity of the LI token. This also provides evidence to receiving entities regarding the identity of the LI gatherer. An LI gatherer may generate several tokens for the same LI, e.g. if an LI subject uses two or more LIPAs. There is also provision for the inclusion of an optional public key certificate for the LI gatherer's public key.
3) **LI token distribution.** When LI is required, an LI token is provided to the LBS provider wishing to use the LI for service provision. This could occur in a variety of ways, e.g. by using third party LI token repositories, by sending the LI token via the UD, or by direct transfer from the LI gatherer to the service provider.
4) **LI token verification and decryption.** Once an LBS provider wishing to use LI receives an LI token, it must submit it to the appropriate LIPA. From the LI token the LBS provider can establish the identity of the LI subject, the identifier for the LI token and the identity of the LIPA, but not the LI or constraints since they are encrypted.

   Upon receiving the LI token, the LIPA verifies the signature, then decrypts the LI and the constraints, and checks if access to this LI is permitted for the requesting LBS provider. If access to the LI is permitted by the constraints, the LIPA returns the LI, the date/time of expiry of the LI, and the identifier of the LI token, all encrypted with the public key of the LBS provider, and signed by the LIPA. If permission is denied, a message stating this, together with the identity of the LI token, is returned to the LBS provider.

There are numerous ways that the LIPA may generate income for the provision of its service. The LIPA may charge for each request for LI which it receives, or each successful request for LI, i.e. when LI is sent to a LBS provider by a LIPA. Also,

billing may be per LI token or per individual request. The entities which could potentially be billed for the LIPA service are the LI subject and the LBS provider. Billing the LI subject may result in a scenario where LBSs could request LI from the LIPA, which will charge the LI subject whether or not the LBS provider gives any service to the subject, and this is clearly not a desirable scenario. Alternatively, billing the LBS provider appears a more appropriate solution since the LBS provider can potentially recover the cost of obtaining the LI by including it in the charge for services provided.

The LI gatherer (unless it is the LI subject him/herself) will also typically require a means of obtaining payment for providing LI tokens. However, the LI gatherer may have no obvious party to charge except for the LI subject. In cases where the LI gatherer provides LI tokens for use by LBS providers not providing services to the LI subject, this is probably unviable. Another possibility might be for the LIPA entities to pass on a percentage of charges they make to LBS providers to the LI gatherers.

B. Requirements for use of the mechanism

This section describes the requirements on the entities involved in use of the mechanism.

The LI gatherer is the entity responsible for creating LI. It must possess a signature key pair. It must also possess a trusted copy of the public encryption key for all the LIPAs used by the LI subjects for which it generates/collects LI. These keys are used to encrypt the LI and the constraints in the LI token. The LI gatherer must also be in possession of a reliable copy of the constraints and LIPA preferences for each LI subject for which it generates LI.

The LIPA entity must possess both a signature key pair and an asymmetric encryption key pair. It must also possess a trusted copy of the verification key of every LI gatherer whose LI it needs to process, and a trusted copy of the public encryption key of each service provider to whom it might wish to provide decrypted LI. (The need for LIPAs to hold public keys of LI gatherers and LBS providers can be obviated by requiring LI gatherers and LBS providers to obtain and distribute public key certificates).

Each LBS provider must possess a trusted copy of the public signature verification key of each LIPA with which it interacts. It must also possess an asymmetric encryption key pair.

It is assumed that all the necessary encryption and signature algorithms have been globally agreed before use of the scheme.

C. LI creation

The entity responsible for generating LI is also responsible for creating what we refer to as an LI token. At the time of creation (or acquisition) of the LI, we suppose that the LI gatherer generates accompanying constraints C based on pre-specified LI subject preferences. The structure of the LI token is described below.

LI Token: $E_{e_L}(LI\|C)\|\ I_L\|I_S\|TokenID\|I_G\|$
$S_G(E_{e_L}(LI\|C)\|I_L\|I_S\|TokenID\|I_G)\|\ [Cert_G]$

where: $E_K(X)$ denotes the asymmetric encryption of data string X using the public key K; $S_A(X)$ denotes a digital signature (not providing message recovery) computed on data string X using the private key of entity A; e_X represents the public encryption key of entity X; $X\|Y$ represents the concatenation of data items X and Y; L represents the LIPA; S represents the LI subject; G represents the LI gatherer; I_X represents an identifier for entity X, e.g. I_G denotes an identifier for the LI gatherer G; $Cert_G$ is the public key certificate of the LI gatherer; $[...]$ represents an optional data item.

The LI token is divided into four parts: the encrypted part, the plaintext part, the digital signature, and the (optional) public key certificate of the LI gatherer. The encrypted section contains the LI and the constraints, C. These are encrypted using the public key of the LIPA, e_L. This ensures that entities other than the LIPA cannot see this information. The plaintext part consists of I_L, I_S, $TokenID$ and I_G. The identifier I_L identifies the LIPA whose public key has been used to encrypt the LI and the constraints. This enables any entity wishing to gain access to the contents of an LI token to determine which LIPA it can be requested from. This identifier could take a variety of forms, e.g. a URL or an IP address. The identifier I_S allows any entity to identify the LI Subject to which the LI in the token relates. This identifier may be a pseudonym. The $TokenID$ is an identifier which, in conjunction with I_G, enables an LI token to be uniquely identified. The identifier I_G allows any entity to determine which entity generated the LI token. This also enables entities to decide which public key to use to verify the digital signature. This identifier may also be a pseudonym. The digital signature is computed over both the encrypted and plaintext parts of the LI token. This provides assurance that the LI Token has not been tampered with, and authenticates the entity which created the LI. The certificate $Cert_G$ may be optionally included in the LI token. This makes it easier for LIPAs which communicate with many LI subjects to obtain the necessary public keys.

Before proceeding, note that the encrypted part of the LI token could alternatively be encrypted using a symmetric encryption scheme with a shared secret key. The major advantage of such an approach would be that a symmetric encryption algorithm is typically much less computationally intensive that an asymmetric scheme. The main disadvantage is the key management overhead, since such an approach would require each LI gatherer to share a secret key with every LIPA with which it 'does business'. A variety of different mechanisms exist to provide the necessary key management functions — see, for example, [17].

D. LI distribution

Section IV-C describes the structure of an LI token. When there is a request for LI or, when an LI subject requests a service, the LI token is sent to the relevant LBS provider.

LI Gatherer $\rightarrow P$:
$E_{e_L}(LI\|C)\|\ I_L\|I_S\|TokenID\|I_G\|$
$S_G(E_{e_L}(LI\|C)\|I_L\|I_S\|TokenID\|I_G)\|\ [Cert_G]$

where:

$A \rightarrow B$ represents the communication of a message from entity A to entity B; and P represents the LBS provider.

LI should always be distributed within an LI token, regardless of who is sending the LI. The message above describes direct communication of the LI token from the LI gatherer to the LBS provider; however, as mentioned earlier, LI tokens may also be distributed via third parties and between LBS providers.

E. LI use

This section describes how an entity uses an LI token. When a LBS provider decides that it want to gain access to the LI within an LI token, it must send the LI token to the LIPA whose identifier is in the token, and hence whose public key was used to encrypt the LI in the token.

$P \rightarrow$ LIPA entity:
$E_{e_L}(LI \| C) \| \ I_L \| I_S \| TokenID \| I_G \|$
$S_G(E_{e_L}(LI \| C) \| I_L \| I_S \| TokenID \| I_G) \|$
$[Cert_G] \| [Cert_P]$

The above indicates that the LBS provider sends the LI token to the LIPA entity. The LBS provider may also optionally include a certificate for its public key, to avoid the need for the LIPA to possess a trusted copy of every LBS provider's public key. When the LIPA receives the LI token, it must first verify the signature and decrypt the enclosed LI and constraints. If the signature is invalid, or the token syntax is not as expected, then the LBS provider must be sent the 'Permission Denied' message (see below). The LIPA must then check that the LBS is permitted by the constraints of the LI subject to receive this LI. The LIPA must also check the authenticity of the LBS provider, which may be based on the certificate provided by the LBS provider. Details of a mechanism to provide this check for authenticity are not discussed further in this document. If the LBS provider is permitted to have access to the LI in the token, then it may be sent. The structure of the message used to send the LI back to P is described below. The LIPA also keeps a record of the LI token and the entity to which it is providing LI.

LIPA entity $\rightarrow P$:
$E_{e_P}(LI \| Expiry \| TokenID)$
$S_L(E_{e_P}(LI \| Expiry \| TokenID))$

The message from the LIPA to the service entity contains two parts: the encrypted part, which contains LI, $Expiry$ and the $TokenID$, and the signature. The encrypted part is encrypted with the public key of the service entity requesting the LI. This ensures that only the service entity can read this information, preventing malicious parties intercepting data while in transit. $Expiry$ is a time-stamp extracted from the constraints and specifies when the LI expires, i.e. when the LI should be deleted. This is the only information from the constraints which needs to be sent to the service entity. The $TokenID$ allows the LI subject to relate the LI received from the LIPA to the LI token from which it has been taken. The digital signature allows the receiving entity to check whether the message has been tampered with during transit.

If the requesting entity is not permitted to have access to the LI in the token then the following $PermissionDenied$ message is sent to the requesting entity:

LIPA entity $\rightarrow P$:
$TokenID \| PermissionDenied$

V. Security analysis

In this section we describe how our mechanism addresses control and privacy issues for LI. We also describe certain remaining issues with the mechanism. These could provide suitable topics for further research.

The primary aim is to provide a mechanism which enables the control of access to LI and constraints, enabling a greater degree of privacy without divulging extra personal information. By enabling the LIPA to make decisions based on constraints, untrusted entities do not gain access to the information found in constraints or LI. However, this does mean that the LIPA has access to both the constraints and the LI. Should the LIPA be compromised, the malicious party would have access to both the LI and the constraints of any LI subject using its services.

Once an entity is in possession of LI, maintaining control of this information is a difficult task. Ensuring that LI is managed according to the preferences of the LI subject once an entity possesses it, can only be based on trust. A problem inherent to LI is that when an entity has plaintext LI, they are free to do with it as they please. Our mechanism aims to provide LI only to entities which can be trusted, giving the LI subject control over their LI. Of course, even trusted entities cannot be trusted all the time and once these trusted entities have this LI, the LI subject can only rely on a regulatory or legal authority to ensure that messages are being transmitted in the manner which has been previously agreed. If an entity wishes to redistribute the LI of an LI subject, it should only distribute the LI token. If it chooses to redistribute LI in other forms, then this can only be addressed by some form of policing, e.g. through peer enforcement. Of course this could enhanced by a regulatory authority which ensures that rules are being adhered to.

Auditability should allow the identification of entities acting in violation of the rules set by the constraints. Identifying these entities is difficult, and is a desirable property. The use of peer pressure to enable auditability was introduced in [3]. To prevent unauthorised distribution of LI, its origin, i.e. the entity responsible for generating the LI token, must be verifiable. In addition, users of LI must be accountable for its use. Therefore, if a malicious entity redistributes LI in a way prohibited by the LI constraints, the recipient will detect this, and the malicious entity can be held responsible for the breach of constraints.

An additional concern is the potential for overloading the LIPA with requests for access to LI. This entity is of course, the central point for LI requests from service providers. This problem can be addressed by distributing the LIPA service across multiple servers, thereby removing the potential bottleneck and the single point of failure.

VI. Conclusion

This paper addresses the issue of control and privacy of LI and associated usage constraints by introducing a Trusted Third Party based framework. We have introduced a mechanism which gives the end user the ability to control their LI without having to divulge additional personal data.

Acknowledgment

The research presented in this paper is being supported by sponsorship from Toshiba Telecommunications Research Laboratory, UK.

References

[1] U. Varshney and R. Vetter, "Mobile commerce: framework, applications and networking support," *Mobile Networks and Applications*, vol. 7, no. 3, pp. 185–198, June 2002.

[2] E. Kaasinen, "User needs for location-aware mobile services," *Personal and Ubiquitous Computing*, vol. 7, no. 1, pp. 70–79, May 2003.

[3] A. S. Gajparia, C. J. Mitchell, and C. Y. Yeun, "Using constraints to protect personal location information," in *Proceedings of VTC 2003 Fall, IEEE Semiannual Vehicular Technology Conference*, vol. 3. IEEE press, 2003, pp. 2112–2116,.

[4] M. Gruteser and D. Grunwald, "Enhancing location privacy in wireless LAN through disposable interface identifiers: A quantitative analysis," in *First ACM international workshop on Wireless mobile applications and services on WLAN hotspots*. ACM Press, September 2003, pp. 46–55.

[5] J. Zao, J. Gahm, G. Troxel, M. Condell, P. Helinek, N. Y. I. Castineyra, and S. Kent, "A public-key based secure mobile IP," *Wireless Networks*, vol. 5, no. 5, pp. 373–390, October 1999.

[6] C.-H. Lee, M.-S. Hwang, and W.-P. Yang, "Enhanced privacy and authentication for the global system for mobile communications," *Wireless Networks*, vol. 5, no. 4, pp. 231–243, July 1999.

[7] G. Myles, A. Friday, and N. Davies, "Preserving privacy in environments with location-based applications," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 56–64, 2003.

[8] T. Aura, M. Roe, and J. Arkko, "Security of internet location management," in *18th Annual Computer Security Applications Conference*. IEEE Computer Society, December 2002, pp. 78–87.

[9] C. Wullems, M. Looi, and A. Clark, "Enhancing the security of internet applications using location: A new model for tamper-resistant GSM location," in *Eighth IEEE International Symposium on Computers and Communications*. IEEE Press, June 2003, pp. 1251–1258.

[10] J. Cuellar, J. Morris, D. Mulligan, J. Peterson, and J. Polk, "Geopriv requirements," IETF, RFC 3693, February 2004.

[11] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *Computer*, vol. 34, no. 8, pp. 57–66, August 2001.

[12] L. Palen and P. Dourish, "Unpacking "privacy" for a networked world," in *Proceedings of the conference on Human factors in computing systems*, G. Cockton and P. Korhonen, Eds. ACM Press, April 2003, pp. 129–136.

[13] A. Adams, "A whole picture is worth a thousand words," *ACM SIGCHI Bulletin*, vol. 35, no. 3, p. 12, May/June 2003.

[14] J. H. Moor, "Towards a theory of privacy in the information age," *ACM SIGCAS Computers and Society*, vol. 27, no. 3, pp. 27–32, September 1997.

[15] L. F. Cranor and B. A. L. Macchia, "Spam!" *Communications of the ACM*, vol. 41, no. 8, pp. 74–83, August 1998.

[16] A. Ranganathan and R. H. Campbell, "Advertising in a pervasive computing environment," in *Proceedings of the second international workshop on Mobile commerce*. ACM Press, September 2002, pp. 10–14.

[17] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*, ser. CRC Press Series on Discrete Mathematics and Its Applications. Boca Raton, Florida: CRC Press, 1997.

An Attack on the Stream Cipher Whiteout

Lillian Kråkmo
Department of Mathematical Sciences
Norwegian University of Science and Technology
Trondheim, Norway
krakmo@math.ntnu.no

Marie Elisabeth Gaup Moe
Centre for Quantifiable Quality of Service in Communication Systems
Norwegian University of Science and Technology
Trondheim, Norway
marieeli@q2s.ntnu.no

***Abstract*— The stream cipher Whiteout is presented and an attack on Whiteout is proposed. This attack involves an active adversary, and recovers 74 bits of the initial states of all LFSRs involved, by using in worst case 30000 ciphertexts of length 2. The attack relies on the fact that the initialization procedure of Whiteout is linear. Whiteout has a secret key of 120 bits and our attack recovers this key by an exhaustive search of only 49 bits.**

***Index Terms*— Stream ciphers, linear keyloading, Whiteout, keystream generator.**

I. Introduction

Linear feedback shift registers (LFSRs) are popular building blocks in pseudorandom bit generators, because of the nice statistical properties of the generated sequences and the practical implementation in hardware. Such generators can be used to produce keystreams for streamciphers. The keystream generator is typically loaded with a secret key which is used for several encryptions, and an initialization vector that is different for each encryption. It is important that the loading procedure of the LFSRs is implemented with care, so that attacks are prevented. In particular the secret key and the initialization vector should not be linearly combined together in the initial state of the generator. In this paper we will demonstrate this by proposing an attack on the keystream generator Whiteout exploiting this weakness. Other attacks exploiting the similarly weak initialization procedure of the stream cipher used in Bluetooth are for example found in [1] and [2].

II. The stream cipher Whiteout

In this section a description of the stream cipher encryption algorithm Whiteout will be given. This algorithm was originally designed for implementation in a dedicated cryptographic chip that could be exported outside NATO countries. However the cipher was never used, discarded, and handed over to us by Thales Communications AS so that we could use it in this study. It should be noted that the cipher is intentionally weak. All detailes about Whiteout given in this section are found in the up to now unpublished specification of Whiteout.

A. Output mode

The key generator of Whiteout shown in Fig. 1 consists of two parts of irregularly clocked, maximal length LFSRs, the block of 3 X-LFSRs and the network of 8 R-LFSRs defined in Table I. These two parts mutually control the clocking of each other via the functions G and F. The G function takes 6 input bits from the X-LFSRs and outputs one bit that decides the start of the path through the network of R-LFSRs that starts in the "node" G and ends up in one of the "leaves" $k0$ or $k1$. Let $x_{i,j}$ denote the bit in position j in the Xi-LFSR. The input bits to the G function are:

$$b_1 = x_{1,8} \qquad b_3 = x_{2,9} \qquad b_5 = x_{3,11}$$
$$b_2 = x_{1,16} \qquad b_4 = x_{2,18} \qquad b_6 = x_{3,22}.$$

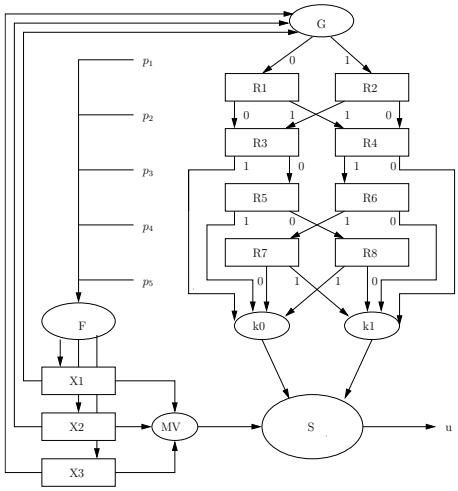


Fig. 1. Whiteout key generator in output mode

TABLE I
Definition of the LFSRs

LFSR	Characteristic polynomial
X1	$1 + x^3 + x^7 + x^{13} + x^{17}$
X2	$1 + x^5 + x^7 + x^{14} + x^{19}$
X3	$1 + x^3 + x^7 + x^{16} + x^{23}$
R1	$1 + x^2 + x^5 + x^8 + x^{11}$
R2	$1 + x^2 + x^6 + x^8 + x^{11}$
R3	$1 + x^2 + x^7 + x^9 + x^{11}$
R4	$1 + x^2 + x^7 + x^{10} + x^{11}$
R5	$1 + x^2 + x^9 + x^{10} + x^{11}$
R6	$1 + x^3 + x^4 + x^{10} + x^{11}$
R7	$1 + x^3 + x^5 + x^7 + x^{11}$
R8	$1 + x^3 + x^5 + x^8 + x^{11}$

TABLE II
H FUNCTION TAPPING POSITIONS

R-LFSR	x	y	z
R1	2	5	7
R2	10	8	3
R3	6	9	4
R4	3	7	8
R5	8	4	2
R6	4	10	6
R7	5	2	9
R8	7	3	5

The output of G is given by:

$$G(b_1, b_2, b_3, b_4, b_5, b_6) = b_2 \oplus b_6 \oplus b_4 b_5 \oplus b_5 b_6 \oplus b_3 b_4 \oplus b_2 b_5 \oplus b_1 b_5 \oplus b_1 b_2 \oplus b_4 b_5 b_6 \oplus b_3 b_5 b_6 \oplus b_3 b_4 b_5 \oplus b_2 b_5 b_6 \oplus b_2 b_4 b_6 \oplus b_2 b_4 b_5 \oplus b_2 b_3 b_6 \oplus b_1 b_3 b_5 \oplus b_1 b_3 b_4 \oplus b_1 b_2 b_5 \oplus b_1 b_2 b_3 b_4 \oplus b_1 b_2 b_4 b_5 \oplus b_1 b_2 b_4 b_6 \oplus b_1 b_3 b_4 b_6 \oplus b_2 b_3 b_4 b_6.$$

The output of G determines which of the LFSRs $R1$ or $R2$ that will be enabled and defines the start of a path through the network of R-LFSRs that will be calculated during each cycle in output mode. The path through the network of R-LFSRs is defined as a binary 5-tuple $(p_1, p_2, p_3, p_4, p_5)$, where p_4 and p_5 may be undefined, p_1 is the output of G, p_2 is the output of $R1$ or $R2$ dependent on which LFSRs that was enabled by p_1, p_3 is the output of $R3$ or $R4$, and so on. The path gives input to the clocking function F and decides which R-LFSRs that will be clocked during the cycle. The path also determines the value of the leaves $k0$ and $k1$. $k0$ and $k1$ are initially set to 0 in the start of each cycle, when the path ends in one of them, the value of the terminating leaf is changed to 1.

The output of the R-LFSRs is given by the function h that takes 3 input bits x, y and z from the R-LFSRs and outputs one bit that determines which R-LFSR next to enable according to Fig. 1. The tapping positions for the input to h for the respective R-LFSRs are given in Table II, the output of h is given by:

$$h(x, y, z) = x \oplus z \oplus xy.$$

The function F decides the clocking of the X-LFSRs and is computed after the path through the R-LFSRs is calculated. F takes 5 input bits and outputs a triple (t_1, t_2, t_3), $t_i \in \{0, 1, 2\}$ where t_i counts the number of steps the LFSR Xi is going to be clocked. Let $r_{i,j}$ denote the bit in position j in the Ri-LFSR. The input bits to F are:

$$c_1 = p_1 \qquad c_4 = r_{5,10} \oplus r_{7,8}$$
$$c_2 = p_2 \qquad c_5 = r_{6,10} \oplus r_{8,8},$$
$$c_3 = p_3$$

where p_1, p_2 and p_3 are the first, second and third bit in the calculated path through the R-LFSRs. From these input bits a corresponding integer value $v \in \{0, 1, \ldots, 31\}$ is calculated as:

$$v = 16c_1 + 8c_2 + 4c_3 + 2c_4 + c_5.$$

TABLE III
F PERMUTATION

v	0	1	2	3	4	5	6	7
u	20	19	17	31	11	8	29	14
v	8	9	10	11	12	13	14	15
u	24	30	15	22	28	25	2	18
v	16	17	18	19	20	21	22	23
u	21	6	13	26	3	23	7	1
v	24	25	26	27	28	29	30	31
u	9	27	12	0	5	4	16	10

TABLE IV
F CLOCKING OF X-LFSRS

u	0	1	2	3	4	5	6	7
$t1$	1	0	1	1	0	0	1	2
$t2$	2	1	0	0	2	1	2	1
$t3$	0	2	2	2	1	2	0	0
u	8	9	10	11	12	13	14	15
$t1$	2	2	2	2	2	1	1	2
$t2$	1	0	1	1	0	2	2	0
$t3$	0	1	0	0	1	0	0	1
u	16	17	18	19	20	21	22	23
$t1$	0	1	1	1	0	2	0	1
$t2$	2	0	2	0	1	1	2	2
$t3$	1	2	0	2	2	0	1	0
u	24	25	26	27	28	29	30	31
$t1$	1	2	2	2	1	0	2	0
$t2$	2	1	1	0	2	1	1	2
$t3$	0	0	0	1	0	2	0	1

F is defined by a permutation of the possible values for v given in Table III, the u-value is then the input to the look-up table given in Table IV, which gives the stepping numbers.

The function MV (majority vote function) outputs the selector bit s:

$$s = MV(x, y, z) = xy \oplus xz \oplus yz,$$

where $x = x_{1,0}$, $y = x_{2,0}$ and $z = x_{3,0}$. If $s = 0$ the produced key bit $u = k0$, else $u = k1$. Since $k0$ and $k1$ always will be the inverse of each other at the time when the key bit is calculated, the produced key bit can be written as

$$u = s \oplus k0.$$

To sum up, one cycle of the key generator of Whiteout in output mode is defined by the following steps:

1) Compute the output of G
2) Compute the path through the network of R-LFSRs
3) Compute MV to find s
4) Compute the keystream bit u

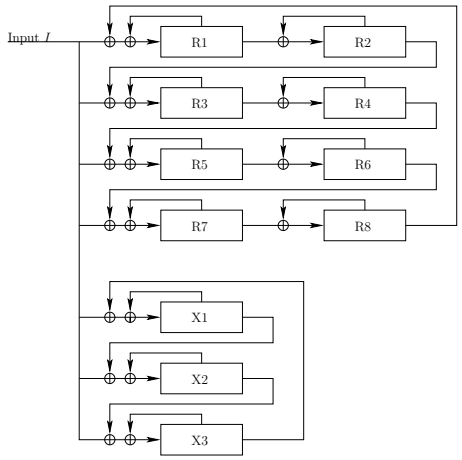


Fig. 2. Whiteout key generator in input mode

5) Compute F to get the stepping of the X-LFSRs
6) Step the X-LFSRs according to computed data
7) Step the R-LFSRs that were enabled in the path.

B. Input mode

In input mode all the LFSRs are connected together as shown in Fig. 2 for loading of the input data consisting of the 120 bit secret key variable (KV) and the 80 bit message indicator (MI). The input mode of Whiteout can also be used for generating a message authentication code (MAC).

Let I be the data input signal, let *fri/fxi* be the internal feedback of the LFSRs Ri/Xi as defined by the characteristic polynomials and let *FRi/FXi* denote the full feedback of the LFSRs in input mode. The feedback bits in input mode are then given by:

$$FR1 = I \oplus fr1 \oplus r_{8,0} \qquad FR2 = fr2 \oplus r_{1,0}$$
$$FR3 = I \oplus fr3 \oplus r_{2,0} \qquad FR4 = fr4 \oplus r_{3,0}$$
$$FR5 = I \oplus fr5 \oplus r_{4,0} \qquad FR6 = fr6 \oplus r_{5,0}$$
$$FR7 = I \oplus fr7 \oplus r_{6,0} \qquad FR8 = fr8 \oplus r_{7,0}$$
$$FX1 = I \oplus fx1 \oplus x_{3,0}$$
$$FX2 = I \oplus fx2 \oplus x_{1,0}$$
$$FX3 = I \oplus fx3 \oplus x_{2,0}$$

The loading of the LFSRs is done in input mode in the following way:

1) All the LFSRs are initially filled with ones
2) Load the 120 bits of KV into the LFSRs starting with the first bit
3) Load 8 zero bits
4) Load the 80 bits of MI starting with the first bit
5) Load 102 zero bits

After the loading procedure the key generator switches to output mode and generates one key bit during each cycle of output mode, that can be used for encryption and decryption. If an LFSR enters output mode in an all-zero state, the feedback bit of this LFSR is forced to be 1 until the LFSR is clocked and this feedback bit enters the register.

The generation of a MAC of a given length n on a given binary message is done in the following way:

1) Load the KV and MI according to the procedure described above
2) Continue loading the message into the LFSRs using input mode
3) Load 102 zeros
4) Switch to output mode
5) The n first produced bits in output mode is the MAC on the message.

III. The initialization procedure

According to the description of Whiteout, the key and the message indicator are loaded into the LFSRs in a linear manner during input mode. In a scenario where only the key is kept secret, this property implies vulnerability to the generator. If we let $(s_0, \ldots, s_{146})$ denote the initial state of the LFSRs as the generator enters output mode, and view the key bits as indeterminants, we can construct a linear system of equations:

$$f_0(x_1, \ldots, x_{120}) = s_0$$
$$\vdots$$
$$f_{146}(x_1, \ldots, x_{120}) = s_{146}.$$

We note that the f_i depend on the message indicator bits. This system can be represented in matrix form by

$$A\mathbf{x} = \mathbf{b},$$

where A is a 147×120 coefficient matrix, $\mathbf{x} = (x_1, \ldots, x_{120})^T$, $\mathbf{b} = (b_0, \ldots, b_{146})^T$, and $b_i = c_i + s_i$, where c_i is the constant term of equation f_i. Consequently, if the adversary succeeds in recovering the initial state of the LFSRs, the key bits can be found by Gaussian elimination of the above system.

We may also consider the situation where parts of the initial state are recovered. More precisely, the adversary has knowledge of $s_I = \{s_i \mid i \in I\}$, where I is a set of indices and $|I| \leq 147$. Let $\mathrm{row}(A, i)$ denote the ith row of A, and let A_I be the matrix formed by only including the rows $\{\mathrm{row}(A, i) \mid i \in I\}$. Furthermore, let $\mathbf{b}_I$ be the corresponding vector of constant terms. In this case, a linear system given by

$$A_I\mathbf{x} = \mathbf{b}_I \tag{1}$$

can be obtained. As usual, $\mathrm{rank}(A_I)$ denotes the number of linearly independent rows of A_I. Clearly, since $\mathbf{x}$ has length 120, $\mathrm{rank}(A_I) \leq 120$. If $\mathrm{rank}(A_I) = 120$, then all of the key bits can be recovered by Gaussian elimination as in the above situation. Generally, A_I has a null space of dimension $120 - \mathrm{rank}(A_I)$, which means that the adversary can recover the key by exhaustive search of $2^{120-\mathrm{rank}(A_I)}$ candidates.

Next we consider a scenario where the adversary has the ability to manipulate the message indicator, i.e. to complement a chosen set of the involved bits. This may be a realistic scenario if the encryption device is physically available to the adversary, for instance as a smart card.

By viewing both the key bits $x_1, \ldots, x_{120}$ and the message indicator bits $m_1, \ldots, m_{80}$ as indeterminants, we can construct the following linear system:

$$\begin{aligned} f'_0(x_1, \ldots, x_{120}, m_1, \ldots, m_{80}) &= s_0 \\ &\vdots \\ f'_{146}(x_1, \ldots, x_{120}, m_1, \ldots, m_{80}) &= s_{146} \end{aligned}$$

Because of the properties of linear functions, this system can be represented in matrix form by

$$A\mathbf{x} + B\mathbf{m} = \mathbf{b}',$$

where B is a 147×80 coefficient matrix, $\mathbf{m} = (m_1, \ldots, m_{80})$, $\mathbf{b}' = (b'_0, \ldots, b'_{146})$ and $b'_i = c'_i + s_i$, where c'_i is the constant term of the function f'_i. In the assumed scenario, the adversary may add an arbitrary vector $\mathbf{m}' \in GF(2)^{80}$ to $\mathbf{m}$, with the effect of complementing the bits of $\mathbf{m}$ corresponding to the elements of $\mathbf{m}'$ equal to 1. The resulting equation is

$$A\mathbf{x} + B(\mathbf{m} + \mathbf{m}') = \mathbf{b}' + B\mathbf{m}' = \mathbf{c}' + (\mathbf{s} + B\mathbf{m}'), \quad (2)$$

where the last equality comes from expressing $\mathbf{b}'$ as $\mathbf{c}' + \mathbf{s}$. Consequently, the adversary may calculate the effect on the initial state bits s_i from adding $\mathbf{m}'$ to $\mathbf{m}$. An interesting question is whether the adversary is able to manipulate the initial state bits in a predefined manner. With this in mind, we consider the equation

$$B\mathbf{m}_i = \mathbf{e}_i, \quad (3)$$

where $\mathbf{e}_i$ is the unit vector $(e_{i,0}, \ldots, e_{i,146})$ with $e_{i,i} = 1$ and $e_{i,j} = 0$ for all j such that $i \neq j$. If such an $\mathbf{m}_i$ is found, replacing $\mathbf{m}'$ by $\mathbf{m}_i$ in (2) gives

$$A\mathbf{x} + B(\mathbf{m} + \mathbf{m}_i) = \mathbf{c}' + (\mathbf{s} + \mathbf{e}_i),$$

meaning that the bit s_i is complemented, while the rest of the initial state bits are unaltered. To predict whether such an $\mathbf{m}_i$ exists for a given i, we simply have to solve the corresponding system of equations. A solution of the equation (3) exists if and only if the vector $\mathbf{e}_i$ belongs to the column space of the matrix B, denoted by $\mathrm{Col}(B)$. The dimension of $\mathrm{Col}(B)$ is upper bounded by 80 so (3) does not have a solution for every $i, 0 \leq i \leq 146$. From this we can conclude that the described technique of complementing bits of the message indicator does not allow the adversary to manipulate every bit of the initial state independently of each other; this can only be done for the s_i such that (3) is consistent. An alternative approach is to look for a subset s_I of initial state bits such that all of the bits in s_I may be complemented independently of each other, without regard to the bits outside the subset. One way of doing this is to search for an s_I such that the corresponding matrix B_I formed by only including the rows $\{\mathrm{row}(B, i) \mid i \in I\}$ has full rank. Let $\mathbf{e_{Ii}}$ be the vector formed from $\mathbf{e}_i$ in an analogous manner, and consider the equation

$$B_I \mathbf{m}_{Ii} = \mathbf{e}_{Ii}. \quad (4)$$

We note that $\mathbf{m}_{Ii}$ has the same number of elements as $\mathbf{m}_i$, with values characteristic for the subset s_I. The advantage of this method is that equation (4) is consistent for every $i \in I$, since $\mathrm{Col}(B_I)$ has full dimension and thus includes $\mathbf{e}_i$ for every i. In other words, the ability to complement the bits in s_I independently of each other is guaranteed without solving equation (4) for every i. Of course, solving the equations is still necessary in order to obtain the vectors $\mathbf{m}_{Ii}$, by which the complementing is performed.

IV. An attack on Whiteout

In this section we sketch an attack on Whiteout, under the assumption that we are able to manipulate the message indicator, i.e. to complement a chosen subset of the involved bits. As suggested by the previous section, the power of the attack depends on our ability to complement the initial state bits in a predefined way. In fact, each procedure of the proposed attack requires that bits can be complemented independently within a specific subset s_I of the initial state bits. This is possible if and only if the corresponding matrix B_I has full rank, which must be verified for all matrices used. Due to this, the procedures of the attack are constructed in an ad hoc manner, based on whether the tested subsets s_I were found to have the required property or not. We have verified that the matrix B has rank 80. This implies that the maximum number of initial state bits which can be complemented independently is 80.

In the following, $f^{(t)}$ is the output function of Whiteout, i.e. the function calculating the keystream bits $u^{(t)}$, given the state of the generator after t time steps. The knowledge of $f^{(t)}$ is critical to the attack. Its algebraic normal form is derived from the description of Whiteout in Section II. For simplicity, we denote the output of the function G at the tth clocking by $g^{(t)}$. The outputs at time t of the function h for the respective LFSRs R1,..., R8 are denoted by $r_1^{(t)}, \ldots, r_8^{(t)}$, the output of the function MV is denoted by $s^{(t)}$, and the value of $k0$ is written as $k0^{(t)}$. For a general binary variable v, $\bar{v} = v + 1$, i.e the complemented v. The keystream bit $u^{(t)}$ is consequently given by

$$\begin{aligned} u^{(t)} &= f^{(t)}(g^{(t)}, r_1^{(t)}, \ldots, r_8^{(t)}, s^{(t)}) \\ &= s^{(t)} \oplus k0^{(t)} \\ &= s^{(t)} \oplus \bar{g}^{(t)} \bar{r_1}^{(t)} r_3^{(t)} \oplus g^{(t)} r_2^{(t)} r_3^{(t)} \\ &\quad \oplus \bar{g}^{(t)} \bar{r_1}^{(t)} \bar{r_3}^{(t)} r_5^{(t)} \oplus g^{(t)} r_2^{(t)} \bar{r_3}^{(t)} r_5^{(t)} \\ &\quad \oplus \bar{g}^{(t)} r_1^{(t)} r_4^{(t)} r_6^{(t)} \bar{r_7}^{(t)} \oplus g^{(t)} \bar{r_2}^{(t)} r_4^{(t)} r_6^{(t)} \bar{r_7}^{(t)} \\ &\quad \oplus \bar{g}^{(t)} \bar{r_1}^{(t)} \bar{r_3}^{(t)} \bar{r_5}^{(t)} r_8^{(t)} \oplus g^{(t)} r_2^{(t)} \bar{r_3}^{(t)} \bar{r_5}^{(t)} r_8^{(t)}. \end{aligned} \quad (5)$$

We note that according to Fig. 1, the 8 terms representing $k0^{(t)}$ correspond to the 8 respective paths leading to $k0$ in the network. In the attack, $f^{(0)}$ is represented by the initial state bits, $s_1^{(0)}, \ldots, s_{146}^{(0)}$, while $f^{(1)}$ is represented by the state bits after the first clocking, denoted by $s_1^{(1)}, \ldots, s_{146}^{(1)}$.

The attack is divided into two stages. During the first stage, we focus on the initial state bits that determine the first output, $u^{(0)}$, which is computed before the LFSRs are clocked for the first time. The inputs to $f^{(0)}$ constitute a relatively small subset of the initial state bits, in which all

the bits can be complemented independently of each other. By complementing these bits in a systematic manner, and observing how $u^{(0)}$ is affected, we are able to recover all the inputs to $f^{(0)}$.

During the second stage, we focus on the initial state bits that are involved in the second output, $u^{(1)}$, which is computed after the first clocking. In principle, the procedure for recovering these bits is the same as for the first stage, but several modifications are required.

We note that since only some of the involved LFSRs are stepped during the first clocking, many of the input bits to $f^{(1)}$ are exactly the same as the inputs to $f^{(0)}$, so the number of new bits to be obtained is strictly limited. In order to recover more of the initial state bits, we may repeat the second stage several times, with an intentionally modified first clocking.

Assume that we are able to recover a subset s_I of the initial state bits. As described in Section III, we may perform Gaussian elimination on the corresponding matrix A_I, and then recover the key by exhaustive search of $2^{120-\mathrm{rank}(A_I)}$ candidates. This constitutes the final part of the attack.

A. Stage 1

We proceed by describing the overall strategy for the first stage of the attack, where we concentrate on the calculation of the first output $u^{(0)}$. The construction of the involved algorithms will be explained in detail later on.

1) Find $g^{(0)}$ by the algorithm Findg.
2) Find the inputs to G, $b_1^{(0)}, \ldots, b_6^{(0)}$ by one of the search trees GTree0 or GTree1 together with Findg.
3) Find the path through the network of R-LFSRs by the search tree FindPath. If the path satisfies a certain criterion, then continue. If not, then repeat the steps 1-3 with a modified message indicator.
4) Find $r_1^{(0)}, \ldots, r_8^{(0)}$ by the search tree RTree.
5) Find the inputs to h for the respective R-LFSRs, $x_{R1}^{(0)}, y_{R1}^{(0)}, z_{R1}^{(0)}, \ldots, x_{R8}^{(0)}, y_{R8}^{(0)}, z_{R8}^{(0)}$, by the search tree hTree together with RTree.
6) Find the inputs to MV, denoted as $x_{MV}^{(0)}, y_{MV}^{(0)}$ and $z_{MV}^{(0)}$, by the search tree MVTree.

We have verified that all the initial state bits involved in $u^{(0)}$ can be complemented independently. More precisely, we have constructed the matrix B_I corresponding to the subset s_I of all the involved bits, and found B_I to have full rank by Gaussian elimination. The corresponding vectors $\mathbf{m}_{Ii}$, which upon addition to the message indicator m complements the initial state bit s_i, are found by solving equations of the form

$$B_I \mathbf{m}_{Ii} = \mathbf{e}_{Ii},$$

as described in the previous section.

The algorithm Findg exploits that, due to the construction of h, the respective variables $r_1^{(0)}, \ldots, r_8^{(0)}$ can be complemented independently by simply complementing $z_{R1}^{(0)}, \ldots, z_{R8}^{(0)}$. This allows us to manipulate $r_1^{(0)}, \ldots, r_8^{(0)}$ in a systematic manner. By observing $u^{(0)}$ after each manipulation, we can determine whether $k0^{(0)}$ has been complemented or not. Findg executes an adapted series of manipulations, such that the value of $g^{(0)}$ can be recovered within a relatively small number of steps.

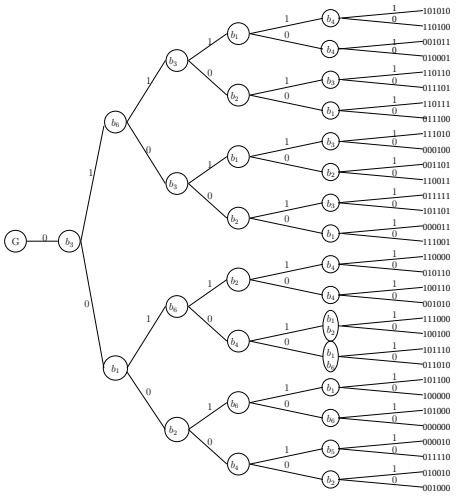


Fig. 3. Searchtree for finding b_1, b_2, b_3, b_4, b_5 and b_6 when the output of G is 0

When $g^{(0)}$ is recovered, the next step is to determine $b_1^{(0)}, \ldots, b_6^{(0)}$. This is done by searching GTree0 or GTree1, according to the value of $g^{(0)}$. The search tree GTree0 is given as Fig. 3. Each node is characterized by one or several of the b_i. During the search, the following procedure is to take place at each node:

1) Complement the $b_i^{(0)}$ corresponding to the b_i of the node.
2) Use Findg to find the value of $g^{(0)}$ resulting from the manipulation of the message indicator.
3) Choose the edge corresponding to the new value of $g^{(0)}$, and move on to the next node. The b_i complemented in step 1 should not be complemented back.

The numbers given at each end node are the respective values of the inputs $b_1, \ldots, b_6$.

The search tree FindPath uses a similar strategy as Findg, but it involves more steps, as it recovers the entire path through the network of R-LFSRs. Due to its size FindPath is not displayed in this paper. We note that since FindPath detects the value of $k0^{(0)}$, it also reveals the value of $s^{(0)}$. The criterion introduced on the path is that it should not involve any of the LFSRs $R5$, $R6$, $R7$ and $R8$. The purpose of this criterion is explained in our discussion of the second stage of the attack. By the assumed attack scenario, we are able to repeat the steps 1-3, with a different MI each time, until the path satisfies the criterion.

The next step is to recover all of the $r_1^{(0)}, \ldots, r_8^{(0)}$, by searching RTree. This tree is based on the output function $f^{(0)}$. To ensure a fairly balanced tree when constructing it, we introduced criteria on the partitions, which were modified in an ad hoc manner, until a complete search tree was obtained. Due to its large size, RTree is not displayed in this paper.

Once every output from h is recovered, the respective inputs $x_{R1}^{(0)}, y_{R1}^{(0)}, z_{R1}^{(0)}, \ldots, x_{R8}^{(0)}, y_{R8}^{(0)}, z_{R8}^{(0)}$ are obtained by searching hTree, given as Fig. 4. During the search, the edges are chosen

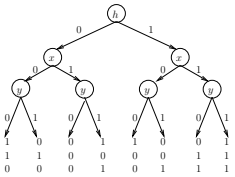


Fig. 4. Searchtree for finding the inputs to the h function

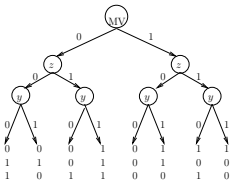


Fig. 5. Searchtree for finding the inputs to the MV function

according to the resulting value of $r_i^{(0)}$, which is recovered by RTree. The numbers given at each end node are the respective values of the inputs $x^{(0)}, y^{(0)}, z^{(0)}$.

Finally, the inputs to MV are found by searching MVTree, given as Figure 5. The edges are chosen according to the value of $s^{(0)}$, which is recovered by observing whether $u^{(0)}$ has been complemented or not. The numbers given at each end node are the respective values of the inputs $x_{MV}^{(0)}, y_{MV}^{(0)}, z_{MV}^{(0)}$.

B. Stage 2

We now consider the second stage of the attack, where we assume to have recovered all the inputs to $f^{(0)}$, and move on to the inputs to $f^{(1)}$. As mentioned earlier, the procedure is basically the same as for the first stage. However, some difficulties are encountered:

1) In order to complement the inputs to $f^{(1)}$, we need to be fully aware of the first clocking, since we need to know which initial state bits the inputs to $f^{(1)}$ are dependent on. However, by studying the function F, we observe that the clocking of the X-LFSRs depends on 3 bits that we have not yet recovered, namely $r_{5,10}$, $r_{7,8}$ and $r_{8,8}$. This difficulty can be avoided by guessing the values of these 3 bits.
2) $u^{(1)}$ depends on the first clocking, in addition to the inputs to $f^{(1)}$, so we need to control a much larger number of bits. Whether this is possible, depends on the nature of the first clocking of the R-LFSRs. This problem can, to some extent, be overcome by introducing a criterion on this clocking.
3) Several of the inputs to $f^{(1)}$ are also involved in the first clocking. Such bits cannot be complemented without potentially affecting the first clocking, which in turn may affect u_1 in an unintended, possibly unpredictable manner. However, if we guess the values of $r_{5,10}$, $r_{7,8}$ and $r_{8,8}$, we assume to know all the bits involved in the first clocking. We are thus able to calculate the unintended effect on the clocking, and if it changes, we may restore the original clocking, by complementing certain other, independent bits.

We proceed by a detailed description of each of the above problems, with the purpose of justifying the suggested solutions.

Due to the nature of the problems 1 and 3, we choose to guess the values of $r_{5,10}$, $r_{7,8}$ and $r_{8,8}$. In the following, we thus assume to be fully aware of the nature of the first clocking, hence the first problem is solved.

The second problem addresses the possibility of dependence between the initial state bits that need to be controlled. In order for the procedure of the first stage to be useful also during the second stage, we need to control the following bits:

- the inputs to $f^{(0)}$ that may affect the clocking, i.e. every input except the inputs to MV,
- the inputs to F not involved in $f^{(0)}$, i.e $r_{5,10}$, $r_{7,8}$ and $r_{8,8}$,
- the inputs to $f^{(1)}$.

It is clear that how many and which input bits we need to control during the second stage, is completely determined by the clocking. We have run a series of tests in order to decide whether the desired control is attainable for different clockings. In particular, we have observed that if $R8$ is stepped during the first clocking, then there is some linear dependency in the subset of initial state bits corresponding to the following variables:

- the inputs to $f^{(1)}$: $x_{R8}^{(1)}$, $y_{R8}^{(1)}$ and $z_{R8}^{(1)}$,
- the inputs to $f^{(0)}$: $x_{R1}^{(0)}, y_{R1}^{(0)}, z_{R1}^{(0)}, \ldots, x_{R8}^{(0)}, y_{R8}^{(0)}, z_{R8}^{(0)}$, $r_{5,10}$, $r_{7,8}$ and $r_{8,8}$.

This may cause problems for several of the procedures involved in the attack, since this dependency prevents us from complementing each of the above bits independently of each other. Due to this, we will prevent that $R8$ is stepped throughout the attack.

Moreover, tests showed that if we allow any combination of $R1$, $R2$, $R3$ and $R4$ to be stepped, and let the X-LFSRs be stepped in an arbitrary way, then the desired control can be obtained. In line with this, we have introduced the criterion that neither one of $R5$, $R6$, $R7$ or $R8$ should be stepped during the first clocking, which is observed to be true with probability $\frac{1}{2}$. Referring to Section III, we thus consider the subset s_I containing all of the bits potentially involved under this criterion, and assume that the vectors m_{Ii} which complements the respective s_i when added to MI, are given by the equation

$$B_I \mathbf{m}_{Ii} = \mathbf{e}_{Ii}. \tag{6}$$

As for the third problem, this needs to be solved separately for each of the relevant bits.

We proceed by giving the overall strategy for the second stage. The starred algorithms are modified versions of the original ones.

1) Find $g^{(1)}$ by the algorithm Findg*.
2) Find the inputs to G, $b_1^{(1)}, \ldots, b_6^{(1)}$ by one of the search trees GTree0* or GTree1* together with Findg*.
3) Find the path through the network of R-LFSRs by the algorithm FindPath*.

4) Find $r_1^{(1)}, \ldots, r_8^{(1)}$ by the search tree RTree*.
5) Find the inputs to h for the respective R-LFSRs, $x_{R1}^{(1)}, y_{R1}^{(1)}, z_{R1}^{(1)}, \ldots, x_{R8}^{(1)}, y_{R8}^{(1)}, z_{R8}^{(1)}$, by the search tree hTree* together with RTree*.
6) Find the inputs to MV, written as $x_{MV}^{(1)}, y_{MV}^{(1)}$ and $z_{MV}^{(1)}$, by the search tree MVTree*.

In Findg*, the bits that may need to be complemented are $z_{R1}^{(1)}, \ldots, z_{R8}^{(1)}$. We note that the $z_{Ri}^{(1)}$ of the R-LFSRs that have not been stepped, have not been involved in the clocking, so they can be complemented during the second stage without causing problems. Furthermore, we observe that neither of the $z_{Ri}^{(1)}$ in the case where Ri has been stepped, has been involved in the clocking. In other words, Findg* works exactly as Findg, when manipulating $z_{R1}^{(1)}, \ldots, z_{R8}^{(1)}$ in the same systematic manner and observing how $u^{(1)}$ is affected.

When searching GTree0* or GTree1*, it may be necessary to complement bits that are also involved in the first clocking, since one of the X-LFSRs has not been stepped. For instance, if $X1$ is not stepped, then complementing $b_1^{(1)}$ or $b_2^{(1)}$ means complementing $b_1^{(0)}$ or $b_2^{(0)}$, respectively. However, since both $X2$ and $X3$ have been stepped, we know that $b_3^{(0)}$, $b_4^{(0)}$, $b_5^{(0)}$ and $b_6^{(0)}$ may be complemented, without affecting anything other than $g^{(0)}$. Hence we may eliminate the effect on $g^{(0)}$ of complementing $b_1^{(1)}$ or $b_2^{(1)}$, by complementing a suitable subset of $\{b_3^{(0)}, b_4^{(0)}, b_5^{(0)}, b_6^{(0)}\}$. We have verified that whenever keeping one of the subsets $\{b_1, b_2\}$, $\{b_3, b_4\}$ and $\{b_5, b_6\}$ fixed, we may always complement g by complementing a suitable subset of the remaining inputs. It is observed that the bit $x_{1,8}$ does not cause any problems, because there is no need to complement it during the second stage; in order to complement $b_2^{(1)}$ if $X1$ is stepped twice, we simply complement one of the other feedback bits.

The algorithm FindPath* complements the same subset of bits as Findg*, so the only modification needed is to change $z_{R1}^{(0)}, \ldots, z_{R8}^{(0)}$ into $z_{R1}^{(1)}, \ldots, z_{R8}^{(1)}$ and $u^{(0)}$ into $u^{(1)}$. The same holds for RTree*. However, when complementing $g^{(1)}$, the $b_i^{(0)}$ should be treated with the same precautions as in Findg*.

When considering hTree*, there are a few bits that must be treated carefully, as it may be necessary to complement all of the bits $x_{R1}^{(1)}, y_{R1}^{(1)}, z_{R1}^{(1)}, \ldots, x_{R8}^{(1)}, y_{R8}^{(1)}, z_{R8}^{(1)}$. We note that again, we only need to consider the bits of the R-LFSRs that have been stepped, since the other ones have not been involved in the clocking. However, there is one exception: The bit $y_{R6}^{(1)}$ involves the initial state bit $r_{6,10}$, which is also involved in the clocking. To deal with this the respective bits may be treated as follows. We have included scenarios where $R5$, $R6$ and $R7$ are stepped, as this will happen during the repetitions of the second stage.

- Assume that $R2$ has been stepped. If we need to complement $x_{R2}^{(1)}$, we simply complement one of the other feedback bits.
- Assume that $R4$ has been stepped, and that we need to complement $y_{R4}^{(1)}$. Then we also complement $z_{R4}^{(0)}$, from which we know that $r_4^{(0)}$ is complemented. By studying the function h, we observe that whenever z is fixed, it is always possible to complement the output by complement x or y or both. Since neither of $x_{R4}^{(1)}$ or $y_{R4}^{(1)}$ are involved in the clocking, we may eliminate the unintended effect on $r_4^{(0)}$, by complementing these bits in a suitable way.
- Assume that $R6$ has not been stepped. If we complement $y_{R6}^{(1)}$, i.e the initial state bit $r_{6,10}$, we also complement the input $c_5^{(0)}$ to the function F, and the clocking may change. This effect may be eliminated by also complementing the other input to $c_5^{(0)}$, the initial state bit $r_{8,8}$. If $R8$ had been stepped, this could be a problem, since $r_{8,8} = x_{R8}^{(1)}$ in this case. However, since $R8$ is never stepped, there is no need to consider this scenario.

As for MVTree*, neither of the inputs $x_{MV}^{(1)}, y_{MV}^{(1)}$ and $z_{MV}^{(1)}$ are involved in the clocking, so we only have to change $x_{MV}^{(0)}, y_{MV}^{(0)}$ and $z_{MV}^{(0)}$ into $x_{MV}^{(1)}, y_{MV}^{(1)}$ and $z_{MV}^{(1)}$, and $s^{(0)}$ into $s^{(1)}$.

Next we consider the repetitions of the second stage, where the clocking is manipulated intentionally, with the purpose of involving more of the initial state bits in the output function.

We have verified that for each of the following clockings, the desired control of the initial state bits can be obtained:

1) Any combination of $R1$, $R2$, $R3$ and $R4$ are stepped, and $X1$, $X2$ and $X3$ are stepped in an arbitrary way,
2) $R1$, $R3$, and $R5$ are stepped, and $X1$, $X2$ and $X3$ are stepped in an arbitrary way,
3) $R2$, $R3$, and $R5$ are stepped, and $X1$, $X2$ and $X3$ are stepped in an arbitrary way,
4) $R2$, $R4$ and $R6$ are stepped, and $X1$, $X2$ and $X3$ are stepped in an arbitrary way.
5) $R1$, $R4$, $R6$ and $R7$ are stepped, and $X1$, $X2$ and $X3$ are stepped in an arbitrary way.

Our knowledge of all the bits determining the clocking and our ability to complement them independently, allows us to manipulate the path through the network of R-LFSRs, in such a way that either one of the cases 1-5 is obtained. We may then repeat the second stage of the attack, and as we know which one of the cases will occur, we may choose in advance the suitable subset s_I of initial state bits to control. We observe that after two suitable repetitions, each of the LFSRs $R1, \ldots, R7$ has been stepped.

The same strategy can be applied to the clocking of the X-LFSRs. By studying the function F and the cases 1-5 above, we observe that we may complement all of the inputs $c_1^{(0)}, \ldots, c_5^{(0)}$, such that all possible clockings of the X-LFSRs are obtainable. As there are 6 different clockings, 5 repetitions are needed, so all in all we need to repeat the second stage $5 - 7$ times.

C. Time complexity of the attack

The time complexity of the first part of the attack, i.e stage 1 and stage 2, including repetitions, will be evaluated in terms of the number of performed bit complementations. To determine this number, we need to investigate the involved algorithms and search trees. We note that when complementing several

bits at a time, this has the same complexity as complementing one bit. We have observed that Findg performs a maximum number of 16 bit complementations and that GTree0 and GTree1 both have depth 16. The actual depth of RTree has not been examined, but the criteria introduced on the partitions imply that it is 17 in the worst case. FindPath has depth 16 and hTree and MVTree both have depth 2. From this we deduce the following worst-case numbers of bit complementations for each of the steps of stage 1:

1) 16
2) $5 \cdot 16$
3) 16
4) 17
5) $8 \cdot 2 \cdot 17$
6) 2

Hence, for stage 1, the total number of bit complementations needed is 403 in the worst case.

As for stage 2, we observe that the worst-case number is the same as for stage 1, since the number of bit complementations in each procedure is essentially the same.

In the worst case, stage 2 must be repeated 7 times. Since we are guessing the values of $r_{5,10}$, $r_{7,8}$ and $r_{8,8}$, we must repeat the first part of the attack at most 8 times. We thus conclude that the worst-case number of necessary bit complementations is totally $403 \cdot 9 \cdot 8 \approx 30000$.

As mentioned introductorily in this section, the final part of the attack involves exhaustive search of the remaining candidate keys. The time complexity of this process clearly depends on the number of initial state bits recovered at this stage.

We note that some of the obtained bits are actually feedback bits. For simplicity, the feedback bit of an LFSR at the tth clocking is marked by the subscript fbt. From the description of the attack, we deduce that the following initial state bits are recovered:

- $r_{1,2}$, $r_{1,3}$, $r_{1,5}$, $r_{1,6}$, $r_{1,7}$, $r_{1,8}$,
- $r_{2,3}$, $r_{2,4}$, $r_{2,8}$, $r_{2,9}$, $r_{2,10}$, $r_{2,fb0}$,
- $r_{3,4}$, $r_{3,5}$, $r_{3,6}$, $r_{3,7}$, $r_{3,9}$, $r_{3,10}$,
- $r_{4,3}$, $r_{4,4}$, $r_{4,7}$, $r_{4,8}$, $r_{4,9}$,
- $r_{5,2}$, $r_{5,3}$, $r_{5,4}$, $r_{5,5}$, $r_{5,8}$, $r_{5,9}$, $r_{5,10}$,
- $r_{6,4}$, $r_{6,5}$, $r_{6,6}$, $r_{6,7}$, $r_{6,10}$, $r_{6,fb0}$,
- $r_{7,2}$, $r_{7,3}$, $r_{7,5}$, $r_{7,6}$, $x_{7,8}$, $r_{7,9}$, $r_{7,10}$,
- $r_{8,3}$, $r_{8,5}$, $r_{8,7}$, $r_{8,8}$,
- $x_{1,0}$, $x_{1,1}$, $x_{1,2}$, $x_{1,8}$, $x_{1,9}$, $x_{1,10}$, $x_{1,16}$, $x_{1,fb0}$, $x_{1,fb1}$,
- $x_{2,0}$, $x_{2,1}$, $x_{2,2}$, $x_{2,9}$, $x_{2,10}$, $x_{2,11}$, $x_{2,18}$, $x_{2,fb0}$, $x_{2,fb1}$,
- $x_{3,0}$, $x_{3,1}$, $x_{3,2}$, $x_{3,11}$, $x_{3,12}$, $x_{3,13}$, $x_{3,22}$, $x_{3,fb0}$, $x_{3,fb1}$.

Although the obtained feedback bits can not be directly identified with initial state bits, they are equally valuable to the attacker. As an example, consider the feedback bit $r_{6,fb0}$, and assume that it has the value 0. This gives

$$r_{6,0} \oplus r_{6,3} \oplus r_{6,4} \oplus r_{6,10} = 0,$$

which allows us to eliminate one of the unknown bits. Hence the recovery of a feedback bit may be regarded as recovering an initial state bit, as long as the feedback bit depends on at least one unknown bit. In our case, this is observed to hold for each of the relevant feedback bits.

From the above we conclude that we have recovered 74 initial state bits. Let s_I denote the subset constituted by these bits. By performing Gaussian elimination on the corresponding matrix A_I, the secret key is obtained by exhaustive search of $2^{120-\mathrm{rank}(A_I)}$ candidates. We have verified that A_I has full rank, so the number of candidates to be tested is 2^{46}. Comparing this number to the number of bit complementations performed in the first part, it is clear that the exhaustive search dominates the total time complexity of the attack.

It must be taken under consideration that we guess the values of the bits $r_{5,10}$, $r_{7,8}$ and $r_{8,8}$. In most cases, guessing the wrong values may be detected at an early stage of the attack. But, in worst case, the attack must be repeated 2^3 times in order to be successful. This gives a time complexity corresponding to exhaustive search of 2^{49} candidates.

For a more detailed description of the attack and all the involved algorithms we refer to our master thesis, which can be obtained upon request to the authors. In our master thesis we also point out some correlations in Whiteout, which could be exploited in a divide and conquer correlation attack [3] on a modified version of Whiteout. It would be interesting to see if these correlations could be used in an optimization of the exhaustive search involved in the attack. In our thesis we have also discussed why the MAC algorithm of Whiteout does not have the computation-resistance property, which is required for a MAC to resist forgery attacks [4].

V. Conclusion

We have studied the stream cipher Whiteout and proposed an attack, that may be carried out by an active adversary, on the cipher which recovers 74 bits of the initial states of all LFSRs involved. In worst case 30000 ciphertexts of length 2 are needed to find these bits. The attack relies on the fact that the initialization procedure of Whiteout is linear. This allows us to construct linear equations relating the initial state with the initialization vector variables and the secret key. Whiteout has a secret key of 120 bits and our attack recovers this key by an exhaustive search of only 49 bits.

Acknowledgment

We would like to thank Leif Nilsen and Thales Communications AS for allowing us to study Whiteout, we also thank our supervisor Kristian Gjøsteen for his help and advice when working with our master thesis.

References

[1] J. D. Golić, V. Bagini, and G. Morgari, "Linear Cryptanalysis of Bluetooth Stream Cipher," in *EUROCRYPT 2002*, ser. Lecture Notes in Computer Science, L. R. Knudsen, Ed. Springer, 2002, pp. 238–255.

[2] S. Fluhrer and S. Lucks, "Analysis of the E_0 Encryption System," in *SAC 2001*, ser. Lecture Notes in Computer Science, S. Vadenay and A. Youssef, Eds., vol. 2259. Springer, 2001, pp. 38–48.

[3] T. Siegenthaler, "Decrypting a Class of Stream Ciphers Using Ciphertext Only," *IEEE Transactions on Computers*, vol. c-34, january 1985.

[4] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.

Upgrade of Bluetooth Encryption and Key Replay Attack

Kaarle Ritvanen and Kaisa Nyberg
Nokia Research Center
Helsinki, Finland
{kaarle.ritvanen,kaisa.nyberg}@nokia.com

***Abstract*— After adoption of the Advanced Encryption Standard (AES), security systems of many information and communication systems are being upgraded to support AES based encryption mechanisms. Also within Bluetooth SIG, discussions about replacing the proprietary Bluetooth encryption algorithm E_0 with a new, possibly AES based algorithm have been initiated. The purpose of this paper is to show that this action alone does not improve the overall security because in the Bluetooth system, an active attacker can set up a previously used encryption key by a replay attack. We analyze the problem and show that it is possible to overcome it. Finally, we present alternative modifications to the Bluetooth Baseband and Profiles specifications to support secure use of two different encryption algorithms.**

***Index Terms*— AES, Bluetooth, E_0, encryption, key replay attack**

I. Introduction

Bluetooth is a wireless communications standard intended to be used in personal area networks. Many handheld devices, such as mobile phones, personal digital assistants, and laptop computers, incorporate a Bluetooth radio to enable low-cost wireless data transmission.

Like all modern radio communications systems, Bluetooth uses an encryption algorithm to conceal the transmitted data from eavesdroppers. The encryption algorithm is a stream cipher called E_0, which is based on Linear Feedback Shift Registers (LFSRs) and a summation combiner [1]. The length of the encryption key can be varied between 8 and 128 bits and there is a negotiation procedure by which it can be agreed on. [2, Part H, Chpt. 4]

The E_0 encryption algorithm is a proprietary algorithm designed for the Bluetooth system. There are incentives to introduce a stronger encryption mechanism to Bluetooth, preferably based on the Advanced Encryption Standard (AES) [3]. Nevertheless, support for E_0 cannot be removed, in order to provide compatibility with legacy devices. Previous attacks on the GSM system by Barkan, Biham, and Keller [4] show that two different encryption algorithms within the same system may interfere in an undesirable manner. In one of the scenarios [4, Sect. 7.1], the attacker can force the victim to reuse a previously used encryption key with the weak algorithm, and then recover the key. Then the attacker can decrypt the previously generated ciphertext, even if strong encryption has been used. We show that such an attack may also be possible in Bluetooth if appropriate counter-measures are not taken. Section IV presents the details on how the attack is carried out in the Bluetooth system. The fundamental cause of the problem is that it is possible to replay encryption keys. In Section IV-D, we present our recommendations for the counter-measures that would be sufficient to allow a second encryption algorithm to be securely taken into use in Bluetooth system.

It should be noted that recovering encryption keys is not the only exploit of the possibility for encryption key replay. For instance, Gauthier presented a key replay attack applicable against the EAP-AKA protocol [5] when a Bluetooth link is used between the victim devices [6].

Before presenting the details of our attack, some background information is covered. Section II reviews the state-of-the-art attacks on E_0. The Bluetooth encryption key exchange and authentication procedures are described in Section III.

II. Status of Encryption Algorithm E_0

In 1999, Hermelin and Nyberg showed how it is possible to recover the initial state of the LFSRs from 2^{64} consecutive keystream bits doing a work of 2^{64} [7]. The amount of work has later been reduced to 2^{61} and the required knowledge of keystream bits to 2^{50} [8]. These attacks exploit linear correlations in the summation combiner. Nevertheless, these attacks are of theoretical nature, since the LFSRs are reinitialized after each packet and the length of the keystream never exceeds 2744 bits[1] [2].

At the moment, algebraic attacks seem to be the most effective attacks on E_0. Krause devised an attack requiring a work of 2^{77} but only 128 consecutive bits of known plaintext [9, Sect. 7]. That amount is eminently realistic for an attacker to obtain but the workload is still prohibitive and equivalent to exhaustive key search of a 78-bit key. Later, Armknecht and Krause showed how to recover the initial state from 2^{23} keystream bits doing a work of 2^{68} [10]. By using a technique called *fast algebraic attack*, which requires some precomputation, the amount of work can be reduced to 2^{55} [11], [12].

The aforementioned attacks concentrate on discovering the initial state of the LFSRs from the keystream bits. However, it

[1]The Bluetooth specifications state that the maximum size of payload is 2745 bits. This maximum is achieved by type DM5 packets with 228-byte payload, which maps to 2745 bits due to error-correcting channel coding. However, encryption is applied before channel coding and therefore the maximal-length keystream is used with type DH5 packets having 343-byte payload, which equals to 2744 bits.

has been proven that having an effective algorithm for initial state recovery yields an effective algorithm for recovering the secret key [13].

According to Armknecht, recovering E_0 keys using present known plaintext attacks would require about 128 GB of memory and 8 MB of keystream. With present computing machinery, it would take at least 159 years to perform the computations. [14]

Even if not breakable in practice, E_0 is of lower security level than AES based stream ciphers are currently believed to be. Therefore, if a second AES based encryption algorithm is specified for Bluetooth, the Baseband or the application profile specifications must ensure that two different encryption algorithms can coexist in Bluetooth without causing vulnerabilities to each other.

III. Procedures for Key Exchange and Authentication

This section presents the Bluetooth encryption key exchange and authentication procedures as defined in [2]. The general order in which the related activities take place is:

1) Change of link key
2) Mutual authentication
3) Encryption key exchange

In Bluetooth networks, there is one *master device*, with the clock of which the other devices synchronize. These other devices are called *slaves*. The security protocols are always performed only between the master and a slave but never between two slaves. They are not symmetric and depend on these roles, as we will see.

Mutual authentication and key exchange are mandatory after link key renewal but they are allowed to happen at any other time too. Link keys are discussed in Section III-A. Section III-B explains how authentication works in Bluetooth and Section III-C shows how encryption keys are agreed on.

A. Link Keys

Link key is a shared secret between the communicating devices. In principle, there are four types of link keys:

- combination keys
- unit keys
- temporary keys[2]
- initialization keys

Unit keys are used by devices with limited memory resources but their use is deprecated and they are ignored in this discussion. Initialization keys are used when pairing devices. The attack presented in Section IV assumes that the devices have already been paired, so initialization keys neither are interesting in this context.

Temporary keys are used in point-to-multipoint configurations. As the name suggests, such configurations are usually relatively short-lived. Applications may make the slaves use a common encryption key derived from this common temporary link key to allow encryption of broadcast traffic. Note that also unicast traffic is encrypted with the common key if a temporary link key has been set up. After the master has finished broadcasting that needs encryption, the slaves can be told to fall back to the previous link keys.

In most cases, the link key is a combination key. According to the specifications, combination keys are *semi-permanent*, in the sense that they can be changed but typically have long lifetimes. In fact, the specification suggests that combination keys can be stored into non-volatile memory and used to authenticate and generate encryption keys for future sessions. So it is reasonable to assume that link keys do not change very often in point-to-point configurations.

[2]In [2], keys of this type are called *master keys*, but this term is a bit misleading. In specifications of some other wireless communications systems, such as those of Wireless Local Area Networks [15], long-term link keys (combination key equivalents) are called master keys.

B. Authentication

Bluetooth uses a special algorithm named E_1 to authenticate other devices. It is based on the SAFER+ block cipher [16]. The inputs to E_1 are:

- current link key
- device address of the claimant
- 128-bit challenge

The challenge is generated by the verifier and sent to the claimant. Both parties run E_1 and the claimant sends the response to the verifier that checks whether the results match. E_1 produces also another result, which is called Authenticated Ciphering Offset (ACO). This 96-bit value is used in key exchange and is discussed in Section III-C.

Authentication always takes place for both directions after the link key has been changed. Also the order is fixed: first the master authenticates the slave and then vice versa. This is also true when a temporary multipoint key is taken into use. It is up to the application whether authentication is performed at other times. These additional authentications do not necessarily have to be mutual. In principle, authentication can be performed arbitrarily many times and in arbitrary order unless the application imposes some restrictions on that.

C. Encryption Key Exchange

E_0 encryption keys are generated by an algorithm called E_3, which produces a 128-bit result. If the encryption key is to be shorter than that, the key is shortened by a binary polynomial modulo operation. The inputs to E_3 are:

- current link key
- 128-bit random number
- Ciphering Offset number (COF)

The random number is generated by the master and is supplied to the slave with the control message that requests starting encryption. The last input, COF, takes one of the following values:

- the device address of the master repeated twice, if the link key is a temporary key, or
- ACO produced by the latest authentication, otherwise.

IV. Active Attack on Strong Encryption Algorithm

Let us now assume that a second alternative encryption algorithm is inserted to the Bluetooth system. Then the support for the original E_0 algorithm will be maintained to ensure backward compatibility. Hence, it is necessary to insert a cipher negotiation mechanism to the Link Manager Protocol (LMP) [2, Part C] so that the devices can agree on a common algorithm. Moreover, it is natural to impose change of encryption key after change of encryption algorithm to prevent the same encryption key from being used with two different algorithms.

We also make the following additional assumptions about how the new feature is used in Bluetooth system. The assumptions are realistic and in accordance with the current specification.

1) The same E_3 algorithm is used to generate the encryption keys for all encryption algorithms. This is reasonable, since most modern block ciphers, such as AES, use 128-bit keys.
2) The application does not restrict the order of execution of authentication procedures.
3) The link key is not changed often (i.e. it remains the same throughout all sessions involved in the attack).

Finally, we make the general assumption that passive wiretapping and recording of Bluetooth communication as well as active "Man-in-the-Middle" impersonation is possible in Bluetooth. In particular, we assume that the attacker can impersonate the master to a slave, and send control messages to the slave. Note that we do not assume that the master can adjust its clock as is required by Gauthier's attack [17, Sect. 2].

We show that if these assumptions hold, then it is possible for an active attacker to force a Bluetooth slave device to reuse a previously used encryption key with an encryption algorithm selected by the attacker. In this manner, a situation is created where the attack of Barkan *et al.* works. This violates the requirement that the different encryption algorithms must not pose any threat to each other.

At first, in Section IV-A we consider a simple case involving only combination-type link keys. In Section IV-B, we show that under certain conditions this attack can be even easier to perform. Section IV-C discusses whether the attack can be extended to sessions containing encrypted point-to-multipoint transmissions.

A. Basic Attack

In case of point-to-point configurations, which we are now considering, the value of ACO is directly used as COF, the input to the encryption key generation algorithm E_3. If authentication is performed for both parties, the ACO produced by the latest authentication is used. Hence the factors that determine the encryption key are:

- current link key
- master-supplied random number
- challenge supplied by the verifier of the last authentication
- device address of the claimant of the last authentication

The attack works as follows. At first, the attacker records a session that is encrypted by using a strong algorithm. Prior to that, he sees the master supply the last authentication challenge, observes the random number attached to the encryption start request, and saves those messages.

Later, at a moment best suitable for him, the attacker becomes active and impersonates the master to the slave. The old link key is used, so there is no need for mutual authentication. Now the attacker runs the negotiation procedure to take the weak encryption algorithm into use. As explained in the beginning of Section IV, a procedure to exchange a new encryption key is performed. It may be possible to use an existing ACO value, as discussed in the next subsection. If a new ACO value is needed, the attacker requests the slave to authenticate itself by sending the previously recorded challenge, as allowed by assumption 2. Being unaware of the real link key, the attacker of course cannot verify the response of the slave, but the result is that the challenge he supplies defines the same ACO, as before. Then the attacker initiates encryption by replaying the random number it recorded from the previous session. The resulting encryption key is identical to the one of the session that the attacker recorded.

It is important to note that if the master is the verifier in the last authentication, the encryption key solely depends on values supplied by him.[3] The slave has then no opportunity to affect the key. This enables the attacker to set up the same encryption key by replaying these values, since by assumption 1 the same E_3 algorithm is used with both encryption algorithms. Now the attacker can try to recover the key by using an attack on the weak algorithm, and then decrypt the ciphertext created using the strong algorithm if he succeeds.

B. Using Existing ACO

A variation of the attack may be possible if the same ACO is allowed to be used for several encryption key computations. If the same ACO were used, COF would remain constant for long periods of time, just like the link key. Then we are again in the situation where the master is the only one who affects the encryption key. The specifications do not forbid reusing ACOs. In fact, they encourage using the same ACO for several key computations in certain situations. When discussing mutual authentication after a temporary key has been distributed, they say [2, Part H, Sect. 3.2.8]:

> The ACO values from the authentications shall not replace the current ACO, as this ACO is needed to (re)compute a ciphering key when the master falls back to the previous (non-temporary) link key.

Therefore, it is highly probable that several implementations do not require a fresh ACO for each encryption key derivation.

[3] Indeed, the encryption key depends on the current link key the attacker does not know. But because of assumption 3, it is constant throughout the attack and in that sense does not affect the encryption key. As regards to the device address, the same holds.

Attacking on such implementations necessitates only replaying the random number input for E_3, not the authentication challenge, thus rendering assumption 2 unnecessary. It is not even necessary for the attacker to know the last challenge, it is required only that the replay takes place when the ACO value is the same as in the recorded session.

C. Point-to-Multipoint Configurations

Let us assume that the application allows the master to make the slaves switch to temporary link and encryption keys, and the attacker has recorded a session that contains such encrypted broadcast episodes. It is clear that the attacker is able to recover such parts of the recorded session that were encrypted using a point-to-point key, since he can replay separately all authentications and key exchanges he has seen. But could the attacker somehow recover broadcast encryption keys too?

Before broadcast encryption can be started, a new temporary link key is created and transmitted to the slaves, in encrypted form of course. But as mutual authentication always occurs after this, there is no way for the attacker to remain undetected, since he does not know the new link key. [2, Part H, Sect. 3.2.8]

However, there can be applications that constantly use the temporary link key. In that case, the temporary key is never relinquished and the attack works well, just like in the point-to-point case. Note that in this case, the attacker need not know the authentication challenge, but can send any plausible value, since COF is derived from the master's address.

D. Possible Counter-Measures

Assumption 3 stated that the link key is not changed often. However, if the specifications dictated that the link key must be changed regularly, that would offer some protection against this replay attack. Replaying the challenge and the random number would no longer yield the same encryption key, had the link key been changed. Moreover, as mutual authentication must always occur after change of link key, changing link keys frequently would certainly offer protection against attacks of this kind. Point-to-multipoint applications constantly switching between combination and temporary group keys naturally use this means of protection.

Another possibility to protect against replay attacks is to make the slave always supply the last challenge. LMP definition rules that the slave supplies the last challenge in mutual authentication after the link key has been changed [2, Part C, Sect. 4.2]. However, this does not by itself prevent the master from initiating new authentication and key exchange procedures immediately after that.

We made the assumption that after each negotiation of encryption algorithm a new encryption key must be exchanged. We assumed that in this process authentication is performed only one way: master authenticates the slave. One might think that requiring mutual authentication would be sufficient to prevent the attacker from replaying an encryption key. However, this is not the case. By impersonating the slave to the real master, the attacker can forward the challenge to the master and get the correct response which it forwards to the slave.

We implicitly assumed that the attacker can freely select the encryption algorithms in protocol negotiation phase. This assumption is based on the fact that currently there is no other integrity protection mechanism than encryption in Bluetooth, and encryption cannot be used before the algorithm has been agreed on. In theory, using message authentication codes based on link keys to protect the negotiation would prevent this attack. However, it would not prevent other types of encryption key replay attacks, such as Gauthier's attack mentioned in Section I.

Another counter-measure that prevents the same encryption key from being used for two different encryption algorithms is to specify a new different E_3 algorithm for each new encryption algorithm. But again, other types of replay attacks would not be neutralized.

V. Conclusion

In this paper we demonstrated that just introducing a second stronger encryption algorithm is not sufficient to upgrade the security level as desired. The root cause of the problem is that it may be possible for a master device to replay authentication and key exchange.

Four alternative approaches to protect against the attack discussed in Section IV were proposed:

- The link key is changed frequently.
- Bluetooth application profiles mandate the slave to provide the last authentication challenge before encryption key derivation *and* forbid using a single ACO to derive several encryption keys.
- Encryption algorithm negotiation is authenticated.
- Different key derivation algorithms are specified for each encryption algorithm.

The first is contradictory to the idea of link keys. According to the specifications, combination keys are semi-permanent, although changing them frequently would really increase security against active attacks. The second approach is neither in line with the specifications, which tell the implementors to store the ACO for future use. The specifications should rather encourage avoiding ACO reuse under all circumstances.

The last two approaches would only thwart the attacks on multiple encryption algorithms presented in this paper. The key replay attack by Gauthier would still remain valid. But either one of the first two counter-measures would also work against that attack, and therefore are the recommended options.

References

[1] R. A. Rueppel. Correlation immunity and the summation generator. In H. C. Williams, editor, *Advances in Cryptology — CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 260–272, Santa Barbara, CA, USA, August 1985. Springer.

[2] Core System Package [Controller volume]. Volume 2 of Specification of the Bluetooth System, Version 1.2. Promoter Members of Bluetooth SIG, November 2003.

[3] Specification for the Advanced Encryption Standard (AES). NIST FIPS Publication 197, November 2001.

[4] E. Barkan, E. Biham, and N. Keller. Instant ciphertext-only cryptanalysis of GSM encrypted communications. In Boneh [18], pages 600–616.

[5] J. Arkko and H. Haverinen. Extensible Authentication Protocol Method for UMTS Authentication and Key Agreement (EAP-AKA). Internet Draft (`draft-arkko-pppext-eap-aka-12.txt`), April 2004.

[6] E. Gauthier. A man-in-the-middle attack using Bluetooth in a WLAN interworking environment. 3GPP TSG SA WG3 Meeting #32, S3-040163, February 2004.

[7] M. Hermelin and K. Nyberg. Correlation properties of the Bluetooth combiner. In J-S. Song, editor, *Proceedings of ICISC '99*, volume 1787 of *Lecture Notes in Computer Science*, pages 17–29, Seoul, South Korea, December 1999. Korea University, Springer.

[8] P. Ekdahl and T. Johansson. Some results on correlations in the Bluetooth stream cipher. In *Proceedings of the 10th Joint Conference on Communications and Coding*, Obertauern, Austria, 2000.

[9] M. Krause. BDD-based cryptanalysis of key stream generators. In L. R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 222–237, Amsterdam, The Netherlands, 2002. Springer.

[10] F. Armknecht and M. Krause. Algebraic attacks on combiners with memory. In Boneh [18], pages 162–176.

[11] N. T. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In Boneh [18], pages 177–194.

[12] F. Armknecht. Improving fast algebraic attacks. In B. K. Roy and W. Meier, editors, *Proceedings of Fast Software Encryption 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 65–82, Delhi, India, February 2004. Springer.

[13] F. Armknecht, J. Lano, and B. Preneel. Extending the framework of the resynchronization attack. In *Proceedings of Selected Areas in Cryptography 2004*, Lecture Notes in Computer Science, Waterloo, Ontario, Canada, August 2004. University of Waterloo, Springer.

[14] F. Armknecht. Algebraic attacks on stream ciphers. Presentation in minisymposium "Secure Crypto for Industry" at ECCOMAS 2004, July 2004.

[15] IEEE Standard for Information Technology; Telecommunications and information exchange between systems; Local and metropolitan area networks; Specific requirements; Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications; Amendment 6: Medium Access Control (MAC) Security Enhancements. IEEE Standard 802.11i-2004, July 2004.

[16] J. L. Massey, G. H. Khachatrian, and M. K. Kuregian. Nomination of SAFER+ as Candidate Algorithm for the Advanced Encryption Standard (AES). 1st AES Conference, Ventura, CA, USA, August 1998.

[17] Notes on Gauthier's replay attack on the UE functionality split scenario. 3GPP TSG SA WG3 Meeting #32, S3-040091, February 2004.

[18] D. Boneh, editor. *Advances in Cryptology — CRYPTO 2003, 23rd Annual International Cryptology Conference, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 2003. Springer.

Preventing Coordinated Attacks via Alert Correlation

J. Garcia*, F. Autrel†, J. Borrell*, Y. Bouzida‡,
S. Castillo*, F. Cuppens‡ and G. Navarro*

*UCCD-UAB, 08193 Bellaterra (Spain)
Email: {jgarcia,jborrell,scastillo,gnavarro}@ccd.uab.es

†ONERA-CERT, 31055 Toulouse (France)
Email: fabien.autrel@enst-bretagne.fr

‡GET-ENST-Bretagne, 35576 Cesson Sévigné (France)
Email: {yacine.bouzida,frederic.cuppens}@enst-bretagne.fr

***Abstract*— When attackers gain access to enterprise or corporate networks by compromising authorized users, computers, or applications, the network and its resources can be used to perform distributed and coordinated attacks against third party networks, or even on computers on the network itself. We are working on a decentralized scheme to share alerts in a secure multicast infrastructure to detect and prevent these kind of attacks. In this paper we present a collaborative framework that performs coordinated attack prevention. The detection and prevention process itself is done by a set of collaborative entities that correlate and assemble the pieces of evidence scattered over the different network resources. We also provide an example of how our system can detect and prevent a coordinated attack to demonstrate the practicability of the system.**

***Index Terms*— Intrusion Detection Systems, Publish-Subscribe Systems, Alert Correlation.**

I. Introduction

Despite the advances in network security technology, such as perimeter firewalls, authentication mechanisms and intrusion detection systems, networked systems have never been more vulnerable than today. The proliferation of Internet access to every network device, the increased mobility of these devices, and the introduction of network-enabled applications have rendered traditional network-based security infrastructures vulnerable to a new generation of attacks. Generally, these attacks start with an intrusion to some corporate network through a vulnerable resource and then launching further actions on the network itself or against third party networks. Once harmless hosts and devices have been compromised, they will become active parts in the deployment of new attacks against other networks if the administrator in charge for these resources cannot effectively disarm them.

The use of distributed and coordinated techniques in these kind of attacks is getting more common among the attacker community, since it opens the possibility to perform more complex tasks, such as coordinated port scans, distributed denial of service (DDoS), etc. These techniques are also useful to make their detection more difficult and, normally, these attacks will not be detected by solely considering information from isolated sources of the network. Different events and specific information must be gathered from all sources and combined in order to identify the attack. Information such as suspicious connections, initiation of processes, addition of new files, sudden shifts in network traffic, etc., have to be considered.

According to [7], we can define the term *attack* as a combination of actions performed by a malicious adversary to violate the security policy of a target computer system or a network domain. Therefore, we can define the *attack detection process* as the sequence of elementary actions that should be performed in order to identify and respond to an attack. An *intrusion detection system* (IDS) is the most important component in performing this process. As mentioned in [10], an IDS has to fulfill the requirements of accuracy (it must not confuse a legitimate action with an intrusion), performance (its performance must be enough to carry out real-time intrusion detection), completeness (it should not fail to detect an intrusion), fault tolerance (the IDS must itself be resistant to attacks) and scalability (it must be able to process the worst-case number of events without dropping information).

In this paper, we present an intrusion detection system which provides a decentralized solution to prevent the use of network resources to perform coordinated attacks against third party networks. Our system includes a set of cooperative entities (called prevention cells) which are lodged inside resources of the network. These entities collaborate to detect when the resources where they are lodged are becoming an active part of a coordinated attack. The main difference between our proposal and other related work is that each node that lodges a prevention cell is expected to be the source of one of the different steps of a coordinated attack, not its destination.

The rest of this paper is organized as follows. Section II presents some related work on the detection of distributed attacks. Our system is presented in Section III and its alert correlation mechanism is introduced in Section IV. The utilization of our system inside a real scenario is described in Section V. Finally, conclusions and further work are placed in the last section.

II. Related Work

Currently, there are a great number of publications related to the design of systems that detect and prevent coordinated and distributed attacks. The major part of them are designed as centralized or hierarchical systems that usually present a set of problems associated with the saturation of the service offered by centralized or master domain analyzers.

As shown in [2], centralized systems, such as DIDS [22] and NADIR [14], process their data in a central node despite their distributed data collection. Thus, these schemes are straightforward as they simply place the data at a central node and perform the computation there. On the other hand, hierarchical approaches, such as GrIDS [23], Emerald [20], AAFID [2] and NetSTAT [25], have a layered structure where data is locally preprocessed and filtered. Although they mitigate some weaknesses present at centralized schemes, they still carry out bottleneck, scalability problems and fault tolerance vulnerabilities at the root level.

In contrast to these traditional architectures, alternative approaches such as Micael [9], IDA [1], Sparta [17] and MAIDS [13], propose the use of mobile agent technology to gather the pieces of evidence of an attack (which are scattered over arbitrary locations). The idea of distributing the detection process to different mobile agents has some advantages regarding centralized and hierarchical approaches. For example, these schemes keep the whole system load relatively low and the consumption of the needed resources takes place only where the agents are running. Furthermore, agents are also able to react very quickly when an intrusion has been discovered.

Mobile agent systems and mobile code may seem to be a promising technology to implement decentralized architectures for the detection of coordinated attacks, but the current systems present very simplistic designs and suffer from several limitations. For instance, in most approaches the use of agent technology and mobility is unnecessary and counterproductive. According to [16], mobile agents are used in these designs simply as data containers, a task that can be performed more efficiently by using a simple message passing. Furthermore, mobile agents introduce additional security risks and cause a performance penalty without providing any clear advantage. None of the proposals based on mobile agent technology seem to have a definitive implementation or any industrial application.

Some message passing designs, such as Quicksand [16] and Indra [15], try to eliminate the need for dedicated elements by introducing a message passing infrastructure. Instead of having a central monitoring station to which all data has to be forwarded, there are independent uniform working entities at each host performing similar basic operations. In order to be able to detect coordinated and distributed attacks, the different entities have to collaborate on the intrusion detection activities and cooperate to perform a decentralized correlation algorithm. These architectures have the advantage that no single point of failure or bottlenecks are inherent in their design.

III. Prevention Cells System

In this section we present the design of a system whose main purpose is to detect and prevent coordinated attacks. By means of a set of entities which will be lodged inside the network, the system will prevent the use of network resources to perform coordinated attacks against third party networks. The aim of this system is not to detect incoming attacks against these entities, but to detect when these nodes are the source of one of the several steps of a coordinated attack and to avoid it.

The design of our system has two main goals. The first one is to obtain a modular architecture composed of a set of cooperative entities. These entities will collaborate to detect when the resources where they are lodged are becoming an active part of a coordinated attack against the network they are located at, or against a third party network. Once an attack has been detected, they must be able to prevent the use of their associated resources to finally avoid their participation on the detected attack. The second goal is to have a complete uncoupled relationship between the different components that are these cooperative entities. Having accomplished this, we will be able to distribute these components according to the needs of each resource we want to disarm.

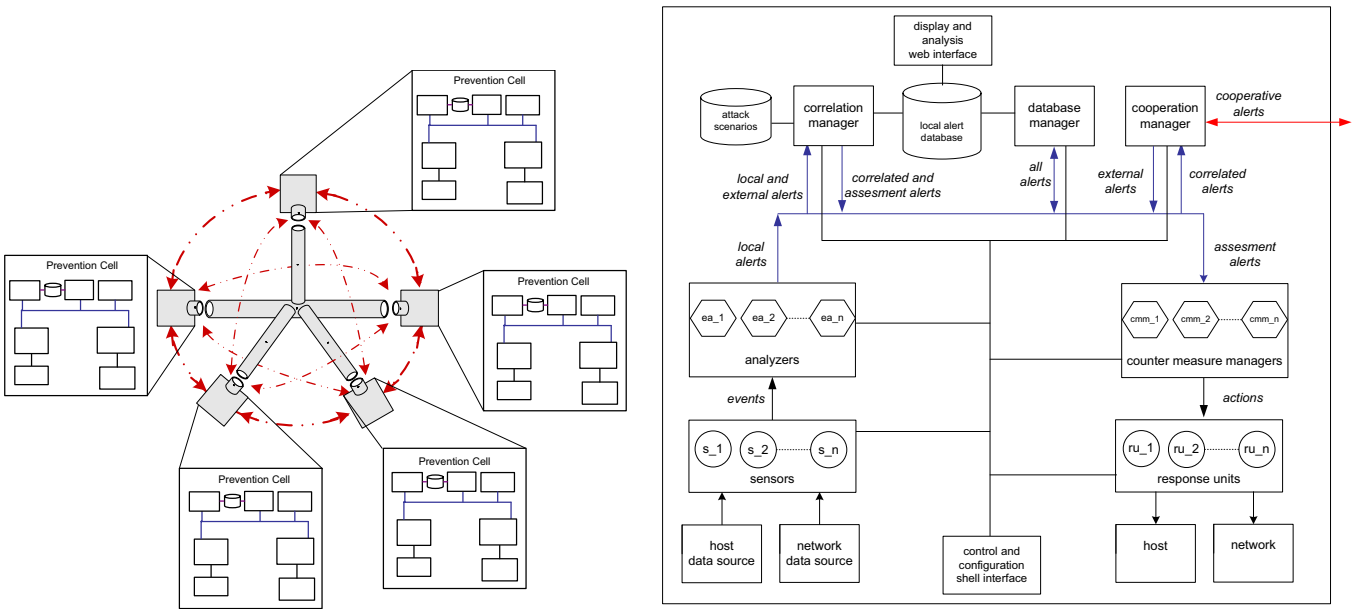
The remainder of this section is organized as follows. First, we present the essential features of the communication architecture of this system and the model used to design it. Then, we describe the elements that make up the different nodes of this architecture.

A. Multicast Communication Architecture

To achieve the first design goal listed above, a multicast architecture is proposed for the communication between the cooperative entities. Through this multicast communication architecture, each one of these entities, called prevention cells, will exchange a set of cooperative messages to collaborate in the decentralized detection process (Figure 1(a)). This architecture must also provide security mechanisms to avoid communication attacks and permit the identification of the different components (like the security mechanisms of the multicast infrastructure introduced in Section VI-D). To do that, we propose the use of a publish-subscribe model.

According to [12], a publish-subscribe system consists of brokers and clients that are connected to brokers. The brokers themselves form the infrastructure used for the routing of notifications. Clients can publish notifications and subscribe to filters that are matched against the notifications passing through the broker network. If a broker receives a new notification it checks if there is a local client that has subscribed to a filter this notification matches. If so, the message is delivered to this client.

The key feature of this model is that components do not know the name or even the existence, of listeners that receive events that they publish. Some other advantages in using a publish-subscribe model for our proposal are the easy implementation of the add and remove operations for components, as much as the introduction of new kind of notifications, the registration of new listeners, and the modification of the set of publishers for a given type of notification.



(a) Message passing architecture based on prevention cells

(b) Basic scheme of a prevention cell

Fig. 1. Collaborative architecture based on prevention cells

B. Prevention Cells

Taking into account the advantages of the publish-subscribe model discussed above, this model is also useful to achieve the independence between components that we have announced as the second goal. Thus, we also propose the use of the publish-subscribe model for the relationship between the internal elements of each prevention cell. By using this model, all of them will be able to produce and consume messages on a secure shared bus.

The internal elements of each prevention cell have been proposed according to the basic components of any IDS, that is, sensors, analyzers, managers, and response units. The messages exchanged between these components are three: *events* (between sensors and analyzers), *alerts* (between analyzers and managers), and *actions* (between managers and response units). These components, and the different messages exchanged between them (Figure 1(b)), are described below:

- *Sensors*, that look for suspicious data on the host or over the network where they are installed and publish this information to a specific event scope (where associated analyzers can subscribe). We propose the use of network based sensors and host based sensors.
- *Analyzers*, that listen to the events published by sensors, to perform a low level correlation process. Thus, these components will consume events and produce local alerts inside the prevention cell. After that, they will publish these alerts at the corresponding scope (the local alert scope).
  We propose the use of misuse based analyzers, with a priori knowledge of sequences and activities of different attacks, and the use of anomaly based analyzers to identify malicious activity comparing the events listened against the representation of normal activities.
- *Correlation manager*, that listens for local and external alerts on their specific scopes and uses the data consumed against its associated coordinated attack scenarios. It will perform a higher correlation process and will be involved in the relative part of the correlation process explained in Section IV. It is also responsible for publishing correlated and assessment alerts.
- *Database manager*, that listens to all of the alert scopes to consume all the alerts produced inside and outside the prevention cell. It will store all these alerts on the local database where it is installed.
- *Cooperation manager*, that listens for cooperative alerts published outside the prevention cell where it is installed and publishes external alerts inside the prevention cell. Furthermore, it also subscribes to correlated alerts and publishes cooperative alerts outside the prevention cell.
- *Counter measure managers*, that listen for assessment alerts published by the correlation manager inside the prevention cell. These managers will be responsible for consuming the assessment alerts and transforming them into the correct actions which will be sent to the associated response units.
- *Response Units*, that take actions produced by their associated counter measure manager to initiate them. Each action is generated to prevent one of the different steps of the detected coordinated attack, and will be performed against the node where the prevention cell is installed. We propose the use of network and host based response units.

IV. Correlation of Alerts

Correlating information held by multiple intrusion detection systems is an approach that has been discussed in several papers. However the goal aimed by those approaches are different and need to be explained.

With the rise of cooperative or distributed intrusion detection frameworks, the problem of reasoning on information coming from multiple sources spread across the monitored system is very important. Correlating this information allows to fulfill different goals, such as information redundancy and scenario detection.

The notion of alert correlation as the process of aggregating alerts related to the same event has been studied in [11], [24], and [4]. They define a similarity relationship between alert attributes to aggregate alerts. The second main approach of alert correlation as the process of detecting scenarios of alerts has been discussed in [19], [5], and [3]. In our proposal we use the latter approach, introducing the notion of alert correlation as the process of finding a set of alerts in the stream of intrusion detection alerts organized into a scenario. Our formalism is explained below.

A. Modelling Actions and Objectives

From the attacker point of view, the attack process can be seen as a planning activity [5]. The intruder can have some knowledge of the system he wants to attack, probably knowing the vulnerabilities present or software and hardware used. If the attacker has a limited knowledge about the targeted system, he can try to gather information by executing actions such as ports scans or using other vulnerability detection tools. Once the attacker has sufficient knowledge of the system to attack, he can define a set of reachable attack objectives.

From the point of view of the victim, those attack objectives constitute a violation of the security policy. In order to reach those attack objectives, the attacker select a set of actions constituting one or multiple scenarios of actions. Finally, from the detection point of view, we want to detect the coordinated attack by constructing scenarios of alerts corresponding to the scenarios of actions executed by the attacker. Hence, we have to model the set of actions available for the attacker and the set of attack objectives. Since we want to react to the detection of ongoing scenarios, we have to model the set of available counter measures.

We use the LAMBDA language [7] to model the actions of the coordinated attacks. LAMBDA provides a logical and generic description of actions, but we use it to model as well the attack objectives and the counter measures. A LAMBDA description of an action is composed mainly of the following attributes:

- *pre-condition*: defines the state of the system needed in order to achieve the action.
- *post-condition*: defines the state of the system after the execution of the action.

Let us consider the modelling of the *BIND Birthday Attack*. This coordinated attack tries to perform a DNS cache poisoning by sending a sufficient number of queries to a vulnerable DNS server based on the BIND software, while sending an equal number of false replies at the same time. A reason for the generation of multiple queries for the same domain name at the same time, could be an attacker trying to hit the needed transaction ID to perform a DNS cache poisoning. Since the transaction ID function is a pseudo-random function, we can supply the brute-force birthday attack based on the birthday paradox.

This attack will result in the storage of an illegal recursive query using the coordination of three techniques. First, a DoS attack to keep the authoritative DNS server from being able to reply. Second, a flooding of queries to an ISP DNS server asking for the IP address of the domain name to be hijacked. And third, a second flooding with the same number of replies formulated by spoofing the IP address of the authoritative DNS server (this way it looks like if these replies were sent from the legitimate nameserver). The attacker avoids the authoritative reply by the first action (the denial of service). If the attack is successful, the targeted ISP DNS will cache the spoofed record for the time indicated in the TTL section of the reply. At this point, the attack is over, but the effect persists for the time the ISP holds the phony record in its nameserver cache. The victim at the ISP is exposed to the spoofed information any time it makes a query for the domain name in question.

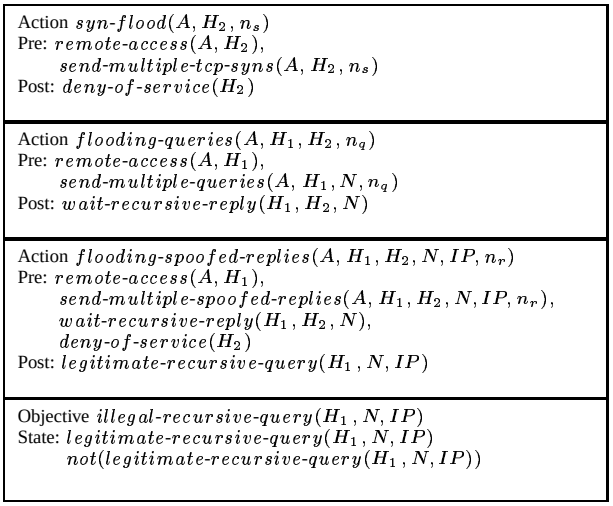


Fig. 2. Modelling the BIND Birthday Attack objective and actions

Figure 2 presents the models for each action that composes the BIND Birthday Attack scenario represented using the LAMBDA language. We also model the attack objective for this scenario as a condition on the system state.

B. Detecting Scenarios

In order to detect the coordinated attack scenario, we use the notion of correlation as the process of finding a set of alerts into the stream of alerts organized into a scenario. To do that, the correlation engine will perform action correlation and alert correlation:

- *Action Correlation* - Two actions A and B are correlated when the realization of A has a positive influence on the realization of B (given that A occurred before B). More formally, if $post(A)$ is the set of post-conditions

of action A and $pre(B)$ is the set of pre-conditions of action B, we say that A and B are *directly correlated* if the following conditions are satisfied:

$\exists E_a$ and E_b such that:

- $(E_a \in post(A) \wedge E_b \in pre(B))$ or $(not(E_a) \in post(A) \wedge not(E_b) \in pre(B))$
- E_a and E_b are unifiable through a most general unifier θ.

Similarly we define the notion of correlation between an action and an attack objective. In this case we correlate the post-condition of an action and the state condition of an objective. The attack objective is modelled as a condition on the system state (Figure 2).

- *Alert Correlation* - Once all the actions available for the attacker have been modeled, we can generate the set of unifiers between all the actions. This generation is done off-line. When an alert is received, we have to bind this alert to an action model and then check for an unifier between the new alert and the already received alerts.

  This set of unifiers is also used to anticipate the possible actions we may see after having observed the beginning of a scenario. Those hypothetic observations are called *virtual actions.*

C. Reacting on Detected Scenarios

Detecting the coordinated attack scenario is interesting but it does not prevent the attacker from reaching his objective. Therefore, we need a mechanism to be able to decide when to execute a counter measure once the scenario has been partially observed and that the next expected action can be blocked through an anti-correlated action:

- *Anti-correlation* - Two actions A and B are anti-correlated when the realization of A has a negative influence on the realization of B (given that A occurred before B). More formally, if $post(A)$ is the set of post-conditions of action A and $pre(B)$ is the set of pre-conditions of action B, we say that A and B are *directly anti-correlated* if the following conditions are satisfied:

  $\exists E_a$ and E_b such that:
  - $(not(E_a) \in post(A) \wedge E_b \in pre(B))$ or $(E_a \in post(A) \wedge not(E_b) \in pre(B))$
  - E_a and E_b are unifiable through a most general unifier θ.

From the modelling point of view, the models for counter measures are not different from the ones representing the set of actions available for the intruder. Actually, a counter measure is an action C anti-correlated with another action A, i.e, one of the predicates in its post-condition is correlated with the negation of one predicate in the pre-condition of action A. This mechanism is provided by the correlation engine through the use of the hypothesis generation mechanism [3]. Each time a new alert is received, the correlation engine finds a set of action models that can be correlated in order to form a scenario leading to an attack objective. This set of hypothesis is then instantiated into a set of virtual alerts. The correlation engine then looks for actions models that can be anti-correlated with the virtual actions. This set of anti-correlated actions becomes the set of counter measures available for the hypothesis represented by the partially observed scenario.

Action $undo\text{-}deny\text{-}of\text{-}service(A, H_1, n_s)$
Pre: $deny\text{-}of\text{-}service(H_1)$,
$send\text{-}multiple\text{-}tcp\text{-}resets(A, H_1, n_s)$
Post: $not(deny\text{-}of\text{-}service(H_1))$

Action $block\text{-}spoofed\text{-}connection(A, H_1, H_2)$
Pre: $spoofed\text{-}connection(A, H_1, H_2)$
Post: $not(spoofed\text{-}connection(A, H_1, H_2))$

Fig. 3. Modelling the BIND Birthday Attack counter measures

Figure 3 presents the models for each action representing the available counter measures for the BIND Birthday Attack scenario. The predicate $not(deny\text{-}of\text{-}service(H_1))$ in the post-condition of action $undo\text{-}deny\text{-}of\text{-}service(A, H_1, n_s)$ is anti-correlated with the predicate $deny\text{-}of\text{-}service(H_1)$. Similarly, the predicate $not(spoofed\text{-}connection(A, H_1, H_2))$ of action $block\text{-}spoofed\text{-}connection(A, H_1, H_2)$ is anti-correlated with the predicate $spoofed\text{-}connection(A, H_1, H_2)$ of attack objective $illegal\text{-}recursive\text{-}query(H_1, N, IP)$.

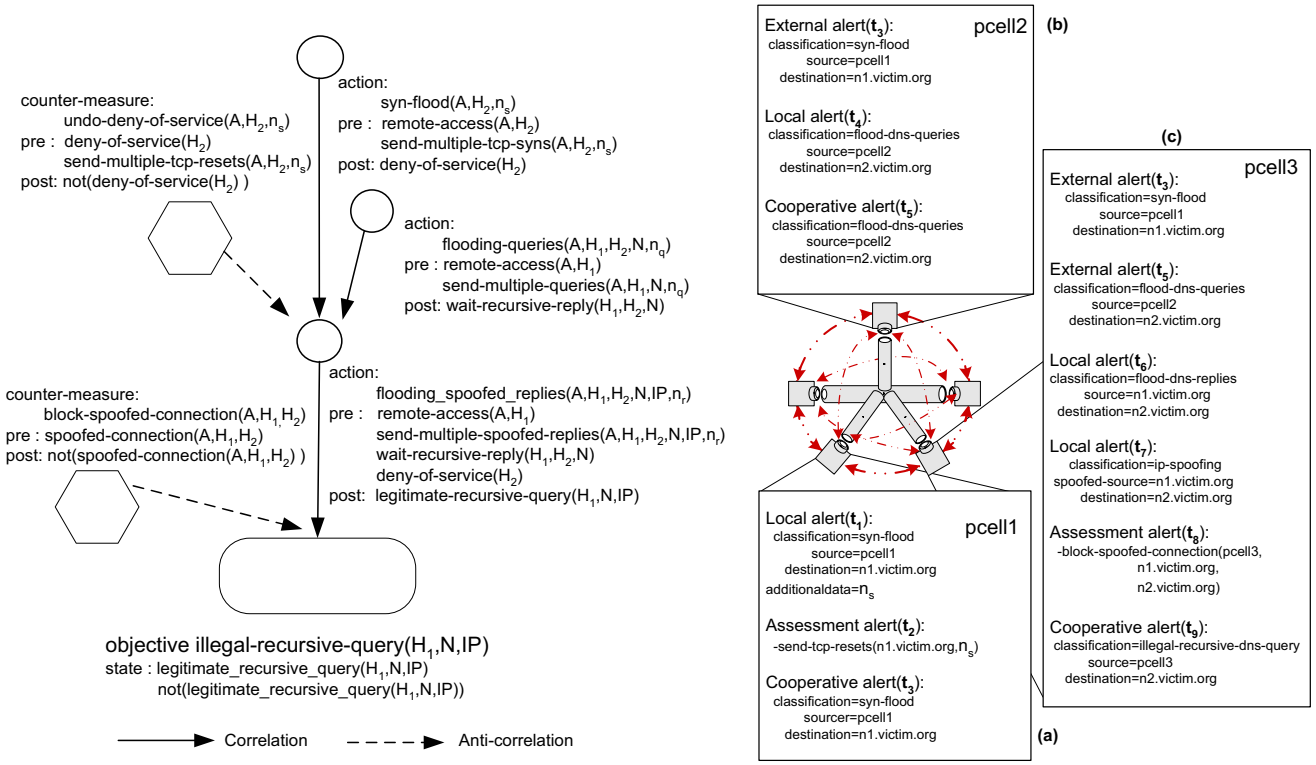
V. Preventing the BIND Birthday Attack

In this section we will discuss the prevention of the BIND Birthday Attack scenario introduced above by using the prevention cells system presented in this paper. This attack is a good example to demonstrate how the components of our architecture handle a possible coordinated attack.

The correlation and anti-correlation graph [6] for this coordinated attack is shown in Figure 4(a). In the first step of this model, A (the agent that performs the whole attack) floods a given host H_2. In the second step, A sends a flood of DNS queries to host H_1 to achieve, that the server process on this host will launch a recursive query to discover the IP address associated to the name N. Then, A starts flooding false recursive replies spoofing the IP address of H_2. Since H_2 is in a mute state, H_1 will never receive the authoritative reply. If one of the false replies has succeeded, H_1 will store the faked information in its cache.

The model of Figure 4(a) proposes two counter measures to prevent the coordinated attack. First, as soon as the host which is performing the SYN flooding DoS against H_2 would detect it, it will neutralize the attack by sending the same number of RST TCP packets to H_2 as SYN TCP packets having received. Second, as soon as the host where the third action (the flooding of spoofed replies to H_1) is detected, it will block these spoofed connections.

To show how the components of our architecture would handle the coordinated attack model described in Figure 4(a), we consider the sequence of alerts described in Figure 4(b). We assume that an attacker targeting the network `victim.org`



(a) Correlation graph for the BIND Birthday Attack

(b) Sequence of alerts raised inside each prevention cell

Fig. 4. Preventing the BIND Birthday Attack by using the prevention cells system

will use resources from another corporate network to perform the coordinated attack. This corporate network is protected with our prevention cells system. The different parts of the attack are detected by three protection cells, named *pcell1*, *pcell2*, and *pcell3* (see Figure 4(b)). For each prevention cell we show the most relevant IDMEF compliant alerts [8] published and consumed by components of the cell. We have simplified quite a lot the information and format of each alert for clarity reasons. For the same reason we assume the correlation and anti-correlation graph for the BIND Birthday Attack is not stored in the attack scenario database of the other prevention cells. Each alert is denoted with ordered identifiers t_i, which correspond to the *DetectionTime* field of the IDMEF alert format.

The first indication of the attack is detected by sensors from *pcell1*. The sensors detect the SYN flooding DoS and generate the local alert t_1. This alert is received by the correlation engine of the cell, which in turn generates the assessment alert t_2 informing that the DoS needs to be neutralized. The assessment alert is observed by the counter measure manager of the prevention cell, which will signal a response unit to block the DoS. Then, by means of the cooperative manager, the prevention cell will send the cooperation alert t_3 to the other prevention cells of the system. This alert is received by the other prevention cells as an external alert notifying that a SYN flooding DoS attack against `n1.victim.org` has been detected and prevented in *pcell1*.

At this point, the prevention cell *pcell1* has prevented the DoS attack against the host `n1.victim.org`, which is the first step of the illegal recursive DNS query scenario. Nevertheless, we cannot ensure that the whole attack is frustrated. It is reasonable to assume that the attacker will try to use another resource not covered by the prevention cells system to commit the final attack. Thus, it is important to try to detect all the steps of the attack and to be able to correlate them in order to identify the whole attack. The next step of the attack, a flooding of DNS queries against `n2.victim.org`, is detected by sensors of *pcell2* that publish it as the local alert t_4. The correlation manager of *pcell2* consumes the alert and produces a corresponding cooperative alert t_5. This alert is sent to the other prevention cells, making them aware that the flooding or DNS queries has been detected in *pcell2*.

Finally, the coordinated attack detection will be completed when the attacker tries the flooding of spoofed replies on the target system (`n2.victim.org`) from the host that lodges the prevention cell *pcell3*. The sensors from *pcell3* detect this flooding and produce the local alerts t_6 and t_7. These alerts, together with the external alerts t_3 and t_5, are correlated by the correlation engine of *pcell3*, resulting in the detection of the coordinated illegal recursive DNS query. This detection step will produce the assessment alert t_8 to block the flooding of spoofed connections. Furthermore, it also involves the production of the cooperative alert t_9 to notify the system that the illegal recursive DNS query scenario has been detected.

VI. Current Development

This section presents a brief overview of an implementation of our prevention system and that deploys all the basic components proposed in this paper. This platform has been developed for GNU/Linux systems in C and C++. Our implementation has been tested on different versions of Linux 2.4.x series and on the versions 2.9.x and 3.x of GNU's gcc compiler. The combination of free high-quality documentation, development and network solutions provided by GNU/Linux operating systems eased the analysis of requirements and the development of this platform. Below, we introduce the main components of our prototype.

A. Sensors and Response Units

Our prototype started with the design and implementation of a set of sensors and response units embedded in the Linux 2.4.x series as kernel modules. Even though, third party sensors and third party response units could easily be integrated in our platform. The implementation of the network sensors and response units is based on the netfilter subsystem, a framework for packet manipulation that enables packet filtering, network address translation and other packet mangling on Linux 2.4.x and upper series.

At this time, we have developed the following network based sensors and response units: a sensor to detect stealth scanning (*syns_s*), a sensor to detect IP spoofing (*spoof_s*), a sensor to detect buffer overflows on the packet payload (*bof_s*), a sensor to detect TCP connection establishments that will be used to infer connection chains (*conn_s*), three sensors to detect denial of service (DoS) attacks based on SYN, UDP and ICMP flooding (*sflood_s*, *uflood_s*, *iflood_s*) and, finally, a response unit capable of producing packet filtering (*pfilter_ru*).

The implementation of the host sensors is based on the interception of some system calls with the purpose of obtaining useful information in the search process of illicit or suspicious activities. On the other hand, the implementation of the host based response units uses the same idea to provide the needed mechanisms to prevent the associated action related with the step of the attack to avoid. We have finished the development of a sensor to monitor the execution of programs (*execve_s*), a sensor to detect which processes want to be finished (*kill_s*) and a response unit able to kill and protect host processes (*kill_ru*).

B. Communication of Events and Actions

The sensors provide the events to each prevention cell analyzer. On the other hand, the counter measure manager of each prevention cell provides the actions to the response units. As we already mentioned, sensors and response units work in kernel space. The complexity of these components and the limitation that supposes to work in a kernel scope entails to design them as daemon processes in user space. Thus, a specific communication mechanism between kernel space and user space is needed.

Among the diverse alternatives for performing the communication between kernel space and user space, we have chosen the *netlink sockets* to bind the proposed sensors and response units with the analyzers and counter measure managers. Netlink sockets is a Linux specific mechanism that provides connectionless and asynchronous bidirectional communication links. Although the use of netlink sockets has been designed with focus on implementing protocols based on IP services, this mechanism can also be used as a standard interface to perform communication between kernel modules and user space processes. Netlink sockets allows us to use the well known primitives from the socket treatment, providing us transparency with the buffering mechanisms.

C. Analyzers and Managers

Both the implementation of the analyzer and the counter measure components, as well as the other managers of each prevention cell, are based on a plug-in mechanism to facilitate the development and the maintenance of the different features that these components will offer. Thus, through the use of Netlink sockets, both the event watcher analyzer and the counter measure manager will consume and produce information.

To generate this information or to manage it, different plug-ins will be enabled or disabled. Some of these plug-ins will be launched in a multi-threading fashion. The event watcher analyzer, for example, will launch the different plug-ins to handle the events received from the sensors using this multi-threading mechanism. This way, it is possible to parallelize the gathering of the different events produced by the set of sensors. Other plug-ins, such as the one responsible for sending actions to the response units, the one responsible for managing external alerts and transform them to internal alerts, etc. will not need the use of this multi-threading mechanism to perform its work.

D. Communication of Alerts

The communication between the analyzers and managers, inside each prevention cell as well as between the other prevention cells of our architecture, is performed by using the Elvin publish-subscribe system [21]. Elvin is a network communication product that provides a simple, flexible and secure communication infrastructure. To be able to use the infrastructure offered by the Elvin publish-subscribe system, both, the analyzers and the managers of our implementation, have been developed using libelvin and e4xx, two portable C and C++ libraries for the Elvin client protocol. Additionally, each host with a prevention cell lodged inside will run an Elvin server to route all the alerts published inside each prevention cell.

Finally, to share the cooperative alerts produced by the different prevention cells in a secure multicast fashion, we use the federation and reliable local-area multicast protocol provided by Elvin and other interesting features offered by this publish-subscribe system, such as fail-over and cryptographic settings. By using SSL at the transport layer we guarantee confidentiality, integrity and authenticity of the cooperative alerts communicated between each prevention cell.

VII. Conclusions and Further Work

We have presented in this paper a decentralized solution for the detection and prevention of distributed and coordinated attacks from network resources. This system uses a secure multicast communication between different entities to avoid their participation in a coordinated attack against third party networks or even the local network. We have also outlined how our system can detect and prevent the BIND Birthday Attack, exploiting the distribution and coordination of the system components. Then, we have briefly discussed the implementation of a platform, which has been developed and which implements the major part of the components of the architecture previously proposed for GNU/Linux systems. Although the detection and reaction components of this platform (sensors and response units implemented as Linux modules) are at this time developed only for Linux 2.4, we plan to upgrade them to Linux 2.6.

As a further work, we are evaluating the possibility to incorporate the formal data model proposed in [18] in our approach. We are also making a more in-depth study of the IDMEF format [8] to solve unnecessary duplicated calculus inside each prevention cell. Finally, we will incorporate intrusion tolerant mechanisms to make our system more reliable when the host that lodges a prevention cell is infected.

Acknowledgments

We would like to thank Michael A. Jaeger for his comments on early drafts of this paper.

The work of J. Garcia, J. Borrell, S. Castillo and G. Navarro has been partially funded by the Spanish Government Commission CICYT, through its grant TIC2003-02041, and the Catalan Government Department DURSI, with its grant 2001SGR-219.

References

[1] M. Asaka, A. Taguchi, and S. Goto. The implementation of IDA: An intrusion detection agent system. In *11th Annual FIRST Conference on Computer Security Incident Handling and Response (FIRST'99)*, 1999.

[2] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, Eugene H. Spafford, and Diego Zamboni. An architecture for intrusion detection using autonomous agents. In *ACSAC 1998*, pages 13–24, 1998.

[3] S. Benferhat, F. Autrel, and F. Cuppens. Enhanced correlation in an intrusion detection process. In *Mathematical Methods, Models and Architecture for Computer Network Security (MMM-ACNS 2003)*, St Petersburg, Russia, September 2003.

[4] F. Cuppens. Managing Alerts in a Multi-Intrusion Detection Environment. In *17th Annual Computer Security Applications Conference New-Orleans*, New-Orleans, USA, December 2001.

[5] F. Cuppens, F. Autrel, A. Miège, and S. Benferhat. Recognizing malicious intention in an intrusion detection process. In *Second International Conference on Hybrid Intelligent Systems (HIS'2002)*, pages 806–817, Santiago, Chile, October 2002.

[6] F. Cuppens, S. Gombault, and T. Sans. Selecting appropriate countermeasures in an intrusion detection framework. In *Proceedings of 17th IEEE Computer Security Foundations Workshop*, Asilomar, Pacific Grove, CA, June 2004.

[7] F. Cuppens and R. Ortalo. LAMBDA: A language to model a database for detection of attacks. In *Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, Toulouse, France, 2000.

[8] D. Curry, H. Debar, and B. Feinstein. Intrusion detection message exchange format data model and extensible markup language (xml) document type definition. Internet draft, January 2004.

[9] J. D. de Queiroz, L. F. R. da Costa Carmo, and L. Pirmez. Micael: An autonomous mobile agent system to protect new generation networked applications. In *2nd Annual Workshop on Recent Advances in Intrusion Detection*, Purdue, IN, USA, September 1999.

[10] H. Debar, M. Dacier, and A. Wespi. *Towards a Taxonomy of Intrusion Detection Systems*. Computer Networks, 1999.

[11] H. Debar and A. Wespi. Aggregation and Correlation of Intrusion-Detection Alerts. In *Fourth International Workshop on the Recent Advances in Intrusion Detection (RAID'2001)*, Davis, USA, October 2001.

[12] D. Garlan, S. Khersonsky, and J. S. Kim. Model checking publish-subscribe systems. In *Proceedings of the 10th International SPIN Workshop*, Portland, Oregon, USA, May, 2003.

[13] G. Helmer, J. Wong, M. Slagell, V. Honavar, L. Miller, R. Lutz, and Y. Wang. Software fault tree and colored petri net based specification, design and implementation of agent-based intrusion detection systems., 2002. Submitted to IEEE Transaction of Software Engineering.

[14] J. Hochberg, K. Jackson, C. Stallins, J. F. McClary, D. DuBois, and J. Ford. NADIR: An automated system for detecting network intrusion and misuse. In *Computer and Security*, volume 12(3), pages 235–248. May 1993.

[15] R. Janakiraman, M. Waldvogel, and Q. Zhang. Indra: A peer-to-peer approach to network intrusion detection and prevention. In *Proceedings of IEEE WETICE 2003*, Austria, June 2003.

[16] C. Kruegel. *Network Alertness - Towards an adaptive, collaborating Intrusion Detection System.* PhD thesis, Technical University of Vienna, June 2002.

[17] C. Kruegel and T. Toth. Flexible, mobile agent based intrusion detection for dynamic networks. In *European Wireless*, Italy, February 2002.

[18] B. Morin, L. Mé, H. Debar, and M. Ducassé. M2D2: a formal data model for intrusion alarm correlation. In *Proceedings of the 5th Recent Advances in Intrusion Detection (RAID2002)*, Zurich, Switzerland, October 2002.

[19] P. Ning, Y. Cui, and D. Reeves. Constructing Attack Scenarios Through Correlation of Intrusion Alerts. In *proceedings of the 9th ACM conference on Computer and communication security*, pages 245–254, Washington DC, USA, 2002.

[20] P. A. Porras and P. G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the 20th National Information Systems Security Conference*, pages 353–365, October 1997.

[21] B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings of the third annual technical conference of AUUG 1997*, pages 243–255, Brisbane, September 1997.

[22] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur. DIDS (distributed intrusion detection system) - motivation, architecture and an early prototype. In *Proceedings 14th National Security Conference*, pages 167–176, October, 1991.

[23] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS – a graph-based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[24] A. Valdes and K. Skinner. Probabilistic alert correlation. In *Fourth International Symposium on Recent Advances in Intrusion Detection (RAID2001)*, pages 58–68, Davis, CA, USA, October 2001.

[25] G. Vigna and R. A. Kemmerer. NetSTAT: A network-based intrusion detection system. *Journal of Computer Security*, 7(1):37–71, 1999.

Mitigating DoS Attacks against the DNS with Dynamic TTL Values

Jarmo Mölsä
Networking Laboratory, Helsinki University of Technology
email: jarmo.molsa@hut.fi

***Abstract*— This paper describes and analyzes a new mechanism to mitigate flooding Denial of Service (DoS) attacks against the Domain Name System (DNS). This mechanism is based on increasing the Time To Live (TTL) value of end-host IP addresses (DNS A records) when a name server is being overloaded with DoS attack traffic. This mechanism is most suitable for popular name servers providing authoritative DNS A records with short TTL values. According to the simulation results, both the average delay and the percentage of failed DNS lookups decrease clearly during a flooding DoS attack. For example, increasing the TTL of DNS A records from 10 minutes to 2 hours decreases the average percentage of failed DNS lookups from 16% to less than 3%, when 90% of the DNS requests are lost due to a DoS attack.**

***Index Terms*— Network Security, Denial of Service, Domain Name System, Time To Live.**

I. Introduction

The Domain Name System (DNS) represents an effective target for Denial of Service (DoS) attacks [1]. In a flooding DoS attack [2][3] a continuous flow of valid-looking DNS requests overloads a network link, a router, a firewall, or a name server. As a result, a legitimate DNS request has problems in reaching a name server and getting an answer. By disabling part of the DNS an attacker is able to prevent or delay access to many services in the Internet. Users typically have only the textual name of a server they are trying to connect to. If the DNS is not available for mapping a textual host name to a numerical IP address, the corresponding server cannot be contacted regardless of the availability of this server. A numerical IP address is always required before it is possible to create a connection to a server.

The objective of this paper is to study how flooding DoS attacks against name servers can be mitigated by modifying the Time To Live (TTL) value of IP addresses (DNS A records). This is an important subject due to the necessary role of DNS in accessing services, due to the prevalence of flooding DoS attacks against name servers [4][5][6], and due to the lack of effective defense mechanisms against these attacks.

The scope of this paper is limited to those name servers providing a final DNS A record (IP address) for a DNS lookup. These name servers are typically the responsibility of the owner of the corresponding *zone* ([7], p. 21). A zone is a non-overlapping part of the DNS. In February, 2003, approximately 68% of the zones in the *com*.-domain were found to be misconfigured [8]. Thus, the level of expertise in operating these name servers is not always high, and a flooding DoS attack against them can easily be successful. Root and Top Level Domain (TLD) name servers, on the other hand, have proved to be very resistant against flooding DoS attacks [9] due to required overprovisioning [10], so they are not considered in this paper.

The main contribution of this paper is to analyze how the total DNS lookup delay and the percentage of failed DNS lookups change when the TTL value of a DNS A record is modified during a flooding DoS attack. The research methodology is based on simulations with the ns-2 network simulator. As another contribution this paper suggests the *dynamic TTL mechanism* to mitigate flooding DoS attacks against name servers of a zone. This mechanism is based on increasing the TTL value of a DNS A record during a flooding DoS attack. The main goal is to increase the cache hit rate at local name servers which reduces the amount of legitimate DNS requests at an overloaded name server. Simulation results show that the mechanism is able to reduce both the average delay associated with the DNS lookup and the amount of completely failed DNS lookups.

At the moment there are no effective defense mechanisms that an organization could use to mitigate flooding DoS attacks against its name servers. Name servers can prevent DNS queries from specific source addresses, but for public services this kind of prevention is not possible. Ingress and egress filtering have been suggested for mitigating flooding DoS attacks using spoofed source IP addresses [4], but these defense mechanisms require extensive deployment in the Internet.

The effect of the length of the TTL on the DNS performance has been studied in [11], in which it was found that the client latency is not as dependent on the use of long TTL values for DNS A records as is commonly believed.

DNS-based load balancing [11][12] and DoS attack resistance require opposite kind of changes to TTL values. This problem manifests itself only during DoS attacks when accurate load balancing must be traded off for the increased availability of a service.

The rest of this paper is structured in the following way. First this paper gives an overview about the operation of the DNS. Then, the dynamic TTL mechanism is described. The next section describes the simulator used to validate the idea. After that the simulation results are shown and explained. The final section concludes the paper.

II. An Overview about the DNS

The Domain Name System (DNS) is used to map domain names in the domain name space into resource records [13][14]. A typical example is to map a textual host name (domain name) into a numerical IP address (type A record).

The *domain name space* is represented as a tree where every node is associated with a *label*. The domain name tree has one leaf node for every end-host accessible from the public Internet (one leaf per end-host name). The internal nodes of the domain name tree reflect hierarchical network domains managed by different organizations. A *domain name* of a node of the tree is written as a sequence of labels from this node towards the root of the tree (e.g. *www.e-service.com.*). If a domain name ends with a dot, it is called a *Fully Qualified Domain Name* (FQDN) which is an absolute and unambiguous name. The last dot in an FQDN marks the root node, i.e. the root node has a null-label.

The DNS is implemented as a distributed data base, where different *name servers* are *authoritative* (responsible) for different non-overlapping *zones* (parts) of the domain name tree. For reliability and performance reasons there are several redundant name servers providing information about a zone.

In the DNS each domain name can be associated with different types of information called *resource records*. From this paper's point of view the most important resource record types are *host address records* (A records) and *authoritative name server records* (NS records). An A record contains a numerical IP address, and an NS record contains a reference to another name server having more detailed information about a domain name to be mapped.

The whole process of translating a domain name into a resource record is called a *name resolution* or a *DNS lookup*. A DNS lookup may require several DNS *requests* and DNS *responses* to be sent. The response messages are either *referrals* or *answers*. A referral contains a list of those name servers having more accurate information. An answer contains the final information requested by an end-host. Naturally a DNS response may indicate an error, like for a non-existent domain name. The result of a complete DNS lookup can be a success (the requested resource record returned), a failure (the requested information not found), or a timeout (no answer within a specified time).

Any name server may cache all received resource records for a period of time defined by the *Time To Live* (TTL) field of a resource record. The TTL is expressed as seconds. The use of caches enhances the performance of DNS.

Without caches every DNS lookup must involve a DNS request to one of the 13 *root name servers* (identified by letters *A* to *M*). In case of a target domain name like *www.e-commerce.com.*, a root name server would return referrals to the *generic Top Level Domain* (gTLD) name servers (identified by letters *A* to *M*) providing authoritative information about the *com.*-domain. A gTLD name server would in turn return referrals to the name servers of the *e-commerce.com.*-subdomain. One of these name servers will then provide the final answer. All these three DNS responses can be stored in caches.

The NS records for the root of the domain name space (IP addresses of the root name servers) are permanently configured into each name server. This guarantees that every name server knows the root of the domain name tree.

A name server can be *iterative* or *recursive*. An iterative name server never sends further DNS requests to other name servers. It simply responds with a referral if it does not know the final answer. A recursive name server will always return the final answer. A recursive name server must typically send DNS requests to several name servers to gradually get closer to the final authoritative answer.

Each end-host must have a *resolver* to access information in the DNS. A resolver is typically implemented as a stub resolver which is simply configured with the IP addresses of the *the local name servers*. Local name servers are recursive and use caches to enhance the DNS performance.

A simple DNS scenario is shown in Fig. 1, which includes a resolver, a set of local name servers, a set of root name servers, a set of gTLD name servers, and a set of name servers in the subdomain of the WWW server. Only local name servers are recursive. All other name servers are expected to be iterative. In this example scenario it does not matter, how many servers there are in any specific name server set. Messages are sent to and handled by one name server in the set. The exact name server can be selected randomly, according to the shortest Round-Trip Time (RTT), etc.

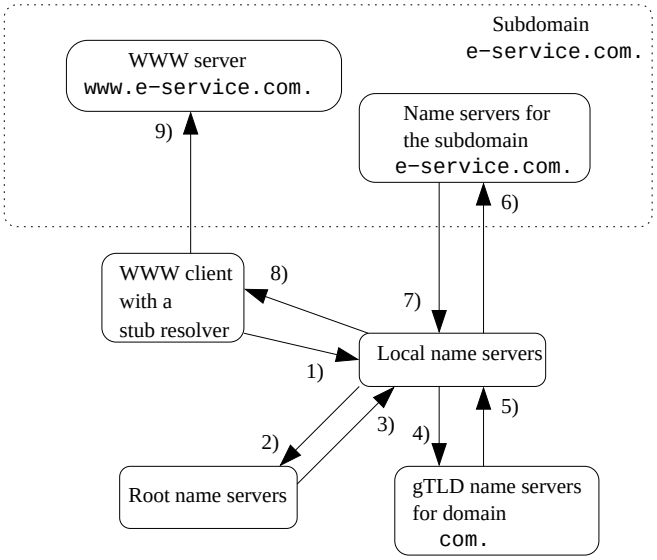


Fig. 1. An example of a DNS lookup.

In the example scenario of Fig. 1 a WWW client wants to connect to the WWW server *www.e-service.com.* The client does not know the IP address of this WWW server so it must use the resolver to send a DNS request to a local name server (message number 1).

The local name server tries to find the following resource records in the following order from the local cache ([13], p. 34): an A record for the *www.e-service.com.*-host, NS records for *e-service.com.*-subdomain, NS records for *com.*-domain and finally the NS records for the root name servers. The first record found defines how the local name server continues.

Either it returns the answer immediately or contacts the closest name server known. The NS records for the root name servers are guaranteed to be found from any cache due to permanent caching of these records from a pre-defined configuration file (hint file).

In this example scenario it is expected that initially the cache contains only the NS records for the root name servers. The local name server must query a root name server (message numbers 2 and 3), a gTLD name server (message numbers 4 and 5) and finally a name server in the subdomain of the WWW server (messages 6 and 7). Messages 7 and 8 include the A record for the WWW server. All DNS responses are cached by the local name server. At the end the WWW client can contact the WWW server by using the numerical IP address in the received DNS answer (message 9).

It should be noted that caching of DNS information is done in other places also. For example, browsers and Java have their own caches. The effect of these kind of separate caches is not included in this paper.

III. The Dynamic TTL Mechanism

This section describes the dynamic TTL mechanism which mitigates flooding DoS attacks against name servers. As a reaction mechanism [15] it is used after a DoS attack is detected manually or automatically (e.g. by inspecting log files or by measuring DNS performance from a remote site). The detection mechanism, however, is not the subject of this paper.

The dynamic TTL mechanism is based on using two different TTL values for each A record: a lower value for normal operation (*default TTL*) and a higher value during a detected DoS attack (*TTL during attack*).

A longer TTL value makes it possible to have higher cache hit rates at remote name servers. When the IP address of a destination host is found from the cache, the overloaded name servers need not be contacted. If the attack is targeted only at the name servers, the final destination can be contacted without problems.

The dynamic TTL mechanism is supposed to be used for A records. NS records are fairly static, and they have a much longer average TTL value (up to several days) than A records.

A major benefit of the dynamic TTL mechanism is that it is easy to implement and does not depend on any third parties. Only local operations are required to mitigate an attack and increase the availability of local services to the public Internet.

IV. The DNS Simulator

The effect of dynamic TTL values in mitigating DoS attacks against name servers was simulated with an OTcl program under the ns-2 network simulator. The setup of client groups, WWW servers, and name servers is shown in Fig. 2. The simulator implements all the basic DNS functions as described earlier in this paper.

A. Clients, WWW Servers and Name Servers

In the simulator there are 200 independent groups of clients. Each client group would reflect, for example, the users of an organization or the customers of a small Internet Service Provider (ISP). Each client group is expected to have one local caching name server. The exact nature of the arrival process of DNS requests at a local name server is not known. Here it is expected that each client group is initiating DNS lookups to the local name server with an exponentially distributed inter-arrival time. Two different average values for the exponentially distributed inter-arrival times were used in the simulations: 120 and 7200 seconds. This makes it possible to study the approximate effect of inter-arrival time on the usefulness of the dynamic TTL mechanism.

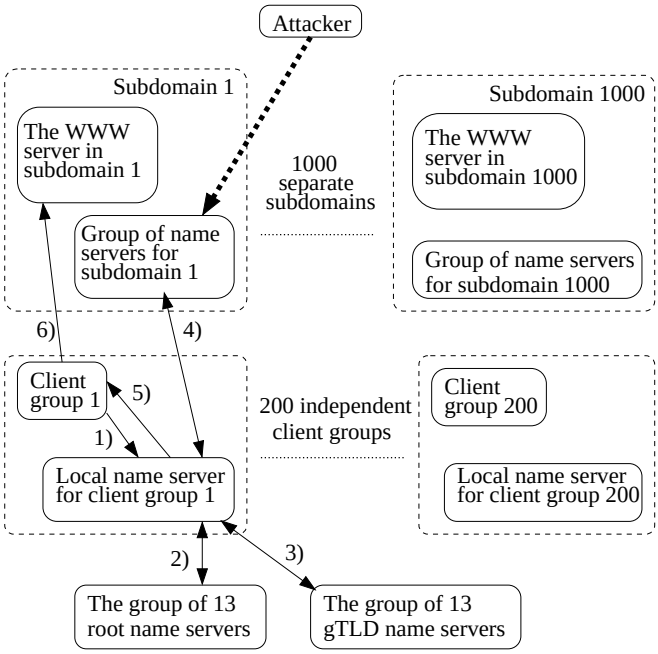


Fig. 2. The simulator setup. 200 independent client groups initiate DNS lookups for 1000 different WWW servers each in a separate subdomain. The WWW servers are ordered according to their Zipf-like popularity, the WWW server in subdomain 1 being the most popular. A flooding DoS attack is targeted against the name servers of subdomain 1 which is using the dynamic TTL mechanism.

The client groups are trying to resolve the name of a server, which is here expected to provide WWW services in a subdomain under the *com.*- or *net.*-domains. The number of WWW servers is 1000, each located in a different subdomain. There are thus 1000 different server subdomains with their own name servers. WWW servers and their corresponding subdomains are identified by their number from 1 to 1000. Each client group chooses the number of the destination WWW server from a Zipf-like distribution, with the parameter α = 0.8. In one study the distribution of web requests was found to be Zipf-like with the value of α being approximately 0.8 [16]. This kind of a distribution means that the WWW server in subdomain 1 is the most popular site, and the WWW server in subdomain 1000 is the least popular site [17].

Each client contains a stub resolver (see [7], p. 26), which is configured to always contact a single local name server. There is one local recursive name server for every client group. Every local name server saves any received resource record in its cache for the duration of the TTL.

Each WWW server subdomain is expected to have a set

of four name servers providing authoritative resource records for the subdomain. The number of these name servers has an effect on the retransmission schedule. Namely, a local name server retransmits in a round-robin fashion to every member of a name server group with an initial timeout length of two seconds ([18], and [7], p. 109).

B. TTL Values

In January, 2004, all root name servers used a TTL value of 172800 seconds (2 days) for the NS records of gTLD name servers. At the same time the gTLD name servers also used a TTL value of 172800 seconds (2 days) for the NS records of subdomain name servers. These TTL values were used in the simulator.

The IP addresses of WWW servers (the final end hosts) use a default TTL of 600 seconds (10 minutes). In one study it was found that the median TTL of A records was approximately 15 minutes when measured from a TTL distribution weighted by access counts [11]. Due to the tendency to use shorter TTL values for A records, the simulator uses 600 seconds as the TTL value for A records.

The dynamic TTL mechanism uses another higher TTL value during DoS attacks. A TTL value of 7200 seconds (2 hours) was chosen for this purpose. This temporary TTL should be in the order of the expected attack length.

C. Delay Distribution for Request-Response Times

The delay between the transmission of a DNS request and the reception of the corresponding response (request-response time) is expected to be normally distributed with a mean value of 92 milliseconds and a standard deviation of 15 milliseconds. In one study [19] it was found that generally no single distribution appears to give a consistently good fit to measured delays of real DNS traffic. A normal distribution with the above mentioned parameter values was found to match reasonably well the request-response times of the L root name server during one measurement period.

The simulator does not take into account the transmission delays between the resolver and the local name server.

D. Flooding DoS Attack

The victims of the DoS attack are the four name servers of the most popular WWW server subdomain having the number 1 in the Zipf-like distribution. The attacker is expected to flood all these name servers with excess traffic, like DNS, ICMP, UDP, or TCP SYN messages. All four name servers of the victim subdomain 1 are attacked in the same way. All remaining name servers (name servers for subdomains 2–1000, root name servers, gTLD name servers) experience no packet-loss and respond always to every DNS request.

The DoS *attack intensity* is defined to be the percentage of lost incoming DNS requests due to the excessive load. For example, only 10% of the DNS requests will be responded, if the attack intensity is 90%.

Random packet-loss typically found in networks is not included in the simulator.

E. Retransmission of Lost DNS Requests

During a flooding DoS attack, only the name servers for subdomain 1 will experience packet-loss. The simulator software includes support for the retransmission of lost DNS requests both at resolvers and local name servers.

The DNS resolver in a client is expected to have a retransmission mechanism similar to the Berkeley Internet Name Domain (BIND) resolver, version 8.2.1 or later ([7], p. 110). The resolver will retransmit only once after a timeout of 5 seconds. If the retransmission is not answered within 10 seconds, the resolver will return an error to the calling software. A resolver will thus spend a maximum of 15 seconds for a DNS lookup.

Local name servers in the simulator have a similar retransmission mechanism as in BIND version 9.2.3 with the following exceptions: Round-Trip Time (RTT) is not calculated and the set of redundant name servers are cycled through only twice. The timeout length is always 2 seconds during these two cycles. A local name server will cease retransmitting after 7 trials.

V. Results of the Simulations

The goal of the simulations is to see how the DNS performance depends on the TTL value of DNS A records during a flooding DoS attack. The simulator provides information about the delay of successful DNS lookups and the proportion of completely failed DNS lookups.

The relevant parameter values in the simulations are the following:

- the length of one simulation is 1 000 000 seconds,
- the DoS attacks starts at the time of 300 000 seconds,
- the attack is carried out at the intensity of 90% (some tests also with the intensity of 50%),
- the default TTL value is 600 seconds (10 minutes),
- the TTL value during a detected DoS attack is 7200 seconds (2 hours), and
- the average time (from exponential distribution) between consecutive DNS lookups from a single client group is either 120 or 7200 seconds (*inter-arrival time*).

All simulation results are calculated from the DNS traffic from any client group (1–200) to subdomain 1, because only subdomain 1 is the target for a DoS attack.

In the simulations it is expected that the DoS attack is detected at the same time when the attack begins (detection is not the subject of this paper). In practice, however, there is always some delay associated with the attack detection. The dynamic TTL mechanism cannot increase the DNS performance until a DoS attack against the name servers is detected and the new TTL values have reached the local name servers.

A. The Delay of Successful DNS Lookups

The average DNS lookup delay is shown in Fig. 3. The X-axis indicates the time in seconds when a DNS lookup was initiated. The Y-axis indicates the delay of DNS lookups averaged over 10 000 second intervals. This figure shows the average delay when the dynamic TTL mechanism is used to

protect the subdomain 1 (*TTL = 600/7200)* and when this mechanism is not used (*TTL = 600*). The results are shown for inter-arrival values of 7200 seconds (thick lines) and 120 seconds (thin lines).

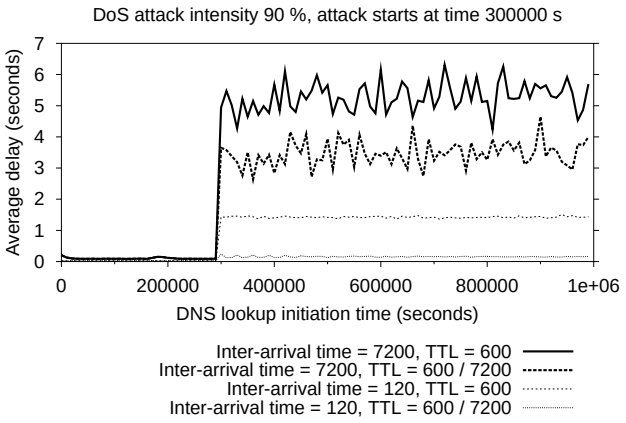


Fig. 3. The delay of successful DNS lookups averaged over 10 000 second intervals.

If the inter-arrival time is 120 seconds, the average delay is reduced almost 90% from 1.5 seconds to approximately 0.16 seconds. When the inter-arrival time is 7200 seconds, the average DNS lookup delay is much longer than when inter-arrival time is 120 seconds. The reason for this is very logical, because the more DNS lookups there are in a time unit, the more lookups will result in a cache hit at the local name server. This pulls down the average lookup delay.

The positive effect of the dynamic TTL mechanism is visible with both inter-arrival times. This effect is, however, stronger when inter-arrival time is shorter. The shorter the inter-arrival time is, the more cache hits will result at the local name server. If a subdomain is visited very seldom, the cached A record will time out before the next visit to it.

As expected, the dynamic TTL mechanism provides the best benefit for those client groups which have many clients referencing a similar set of popular destinations. This increases the possibility for a cache hit at the local name server.

Averaging DNS lookup delays over 10 000 second intervals hides some details of shorter time scale. For this reason the time range from 290 000 seconds to 350 000 seconds of Fig. 3 is magnified in Fig. 4 which shows the delay of DNS lookups averaged over 200 second intervals. Only results for the inter-arrival time of 120 seconds are shown in this figure. Figure 4 shows well that all client groups are practically synchronized due to the short TTL value (600 seconds) after the DoS attack starts at the time of 300 000 seconds. The client groups gradually desynchronize due to the randomness in both the DNS lookup initiation process and the request-response times. Figure 4 also shows the delay until the dynamic TTL mechanism begins to enhance the DNS performance after the attack is detected (the high peak at the time of 300 000 seconds).

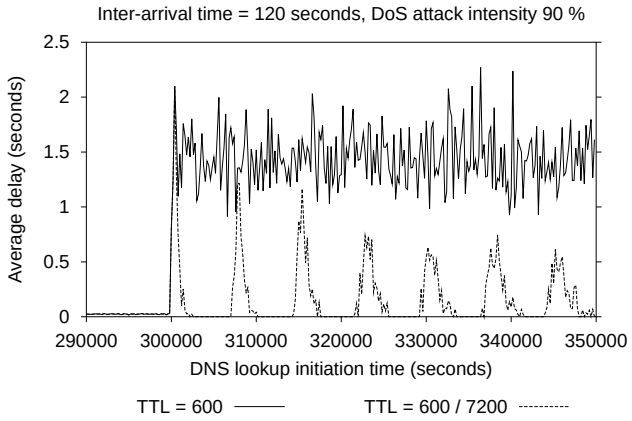


Fig. 4. The delay of successful DNS lookups averaged over 200 second intervals during the time range from 290 000 to 350 000 seconds.

B. The Percentage of Failed DNS Lookups

The average percentage of failed DNS lookups is shown in Fig. 5. The X-axis indicates the time in seconds when a DNS lookup was initiated. The Y-axis indicates the percentage of failed DNS lookups averaged over 10 000 second intervals. A DNS lookup fails if it times out completely at a resolver without an answer. This figure shows the average percentage of failed DNS lookups when the dynamic TTL mechanism is used to protect the subdomain 1 (*TTL = 600/7200)* and when this mechanism is not used (*TTL = 600*). The results are shown for inter-arrival values of 7200 seconds (thick lines) and 120 seconds (thin lines).

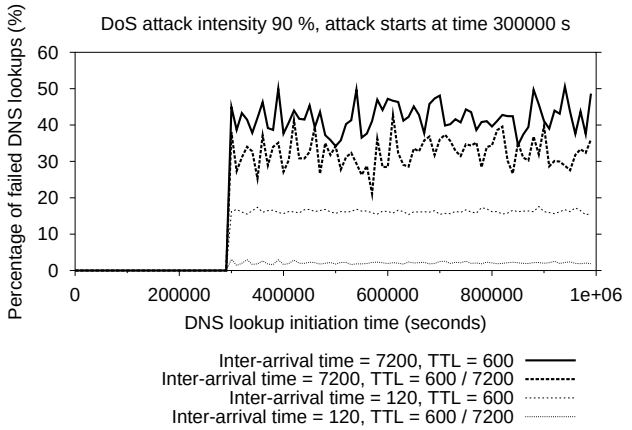


Fig. 5. The percentage of failed DNS lookups averaged over 10 000 second intervals.

The positive effect of the dynamic TTL mechanism is visible with both inter-arrival times. The shorter inter-arrival time (120 seconds) results in more cache hits at the local name server, which decreases the need to send any DNS requests to the overloaded name servers. This reduces the average DNS lookup failure percentage. Increasing the TTL value of A records from 10 minutes to 2 hours decreases the average percentage of failed DNS lookups from 16% to less than 3%, when 90% of the DNS requests are lost due to a DoS attack (inter-arrival time being 120 seconds).

C. Cumulative Distribution Functions for the DNS Lookup Delay

The Cumulative Distribution Functions (CDF) were calculated for several cases with different parameter combinations. The CDF is defined as follows: $CDF(X) = Probability(delay \leq X)$. These CDFs are shown in Fig. 6, where the inter-arrival time is 7200 seconds. Only successful DNS lookups are included here. Because a resolver will timeout completely after 15 seconds, the delay for all successful DNS lookups is less than 15 seconds.

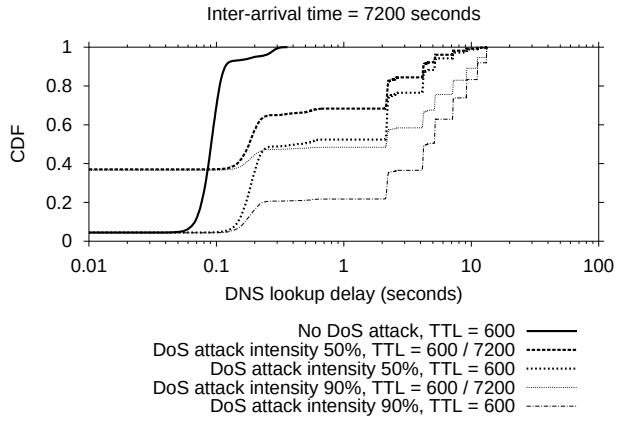


Fig. 6. The Cumulative Distribution Functions (CDF) for the DNS lookup delay, when the inter-arrival time of DNS requests at every local DNS server is 7200 seconds.

As can be seen from the Fig. 6 the dynamic TTL mechanism results in a better CDF, i.e. the mechanism increases the probability of low delays, and decreases the probability of longer delays.

The DNS retransmission policy is visible in these curves. The local name server will retransmit at times of 2 and 4 seconds. At the time of 5 seconds the resolver will timeout and retransmit. After that the local name server will again retransmit with a 2 second interval until the DNS lookup completely times out at the resolver at the time of 15 seconds.

D. The Effect of the Dynamic TTL Mechanism on the DNS Performance

The performance of the DNS during a flooding DoS attack depends on the TTL value. The longer the TTL value, the better the performance. The DNS performance as the function of the TTL value during an attack is shown in Fig. 7. The default TTL is 10 seconds when no DoS attack is present. The thick lines indicate the average delay of successful DNS lookups as a function of the TTL (left Y-axis). The thin lines indicate the percentage of failed DNS lookups as a function of the TTL (right Y-axis). Inter-arrival times of 120 and 7200 seconds were used in these simulations. The DoS attack intensity was 90%.

As can be seen from the Fig. 7 the shorter the inter-arrival time, the higher the performance gain from increasing the TTL. When the inter-arrival time at client groups is 120 seconds, a 50 second TTL (default TTL multiplied by 5) will increase the performance by approximately 10% at the most popular destination domain, and a 400 second TTL (default TTL multiplied by 40) will increase the performance by approximately 50%.

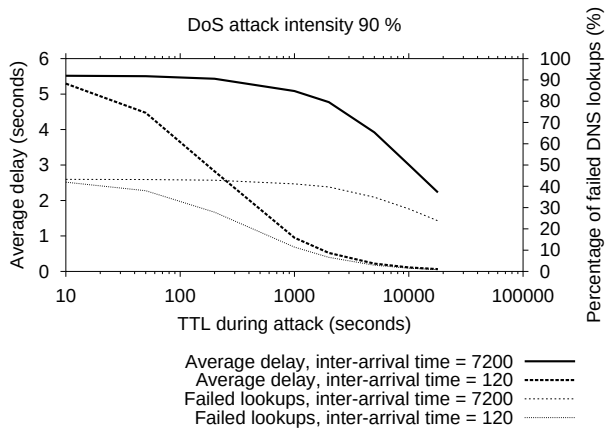


Fig. 7. The effect of the dynamic TTL mechanism on DNS performance. Thick lines indicate average delay of successful DNS lookups (left Y-axis). Thin lines indicate percentage of failed DNS lookups (right Y-axis). Attack intensity is 90%.

DNS-based load balancing depends on a small TTL value for A records. Even though the dynamic TTL mechanism requires relatively long TTL values during an attack, this mechanism can increase the availability of load balanced services. Without any TTL modification many requests for a popular service would fail at the DNS lookup phase, and even a perfect load balancing has no possibility for increasing the DNS performance. The dynamic TTL mechanism will increase the availability at the price of less effective load balancing. This should be seen as a good trade-off. In IPv6 networks one possibility to solve this problem with DNS-based load balancing (application-level anycasting) is to use network-level anycasting [20].

VI. Conclusion

The DNS is a necessary prerequisite for accessing practically any service in the Internet. This makes the DNS an attractive target for attackers who can, for example, disable part of the DNS by flooding a set of name servers with valid-looking but unnecessary DNS requests.

At the moment there are no effective mechanisms to mitigate flooding DoS attacks against name servers providing DNS A records. Existing misconfigurations in most of these name servers make them even more attractive for attackers. Root and gTLD name servers, on the other hand, have proved to be very resistant against these kind of attacks due to required overprovisioning and good expertise. New defense mechanisms are thus required especially for name servers providing DNS A records, i.e. final answers to DNS lookups.

This paper described how dynamic TTL values can be used to mitigate flooding DoS attacks against name servers providing DNS A records. When the name servers of a DNS zone are flooded with unnecessary traffic, increasing the TTL of address resource records increases the cache hit rate at legitimate clients and reduces the amount of DNS traffic against overloaded name servers.

The simulation results clearly show the benefits of this new simple mechanism. For example, when the inter-arrival time of DNS requests is 120 seconds and the attack intensity is 90%, increasing the TTL from 600 seconds to 7200 seconds during an attack reduces the average DNS lookup delay by 90% from approximately 1.5 seconds to 0.16 seconds. The average percentage of failed DNS lookups was also reduced approximately from 16% down to less than 3%. According to the simulation results the modification of the TTL of A records is a useful mechanism for mitigating flooding DoS attacks against the DNS.

REFERENCES

[1] A. Chakrabarti and G. Manimaran, "Internet infrastructure security: A taxonomy," *IEEE Network*, vol. 16, no. 6, pp. 13–21, 2002.

[2] R. K. Chang, "Defending against flooding-based distributed denial-of-service attacks: A tutorial," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 42–51, Oct. 2002.

[3] V. Paxson, "An analysis of using reflectors for distributed denial-of-service attacks," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 3, July 2001.

[4] CERT Coordination Center, "CERT incident note IN-2000-04, denial of service attacks using nameservers," Apr. 2000.

[5] D. Moore, G. M. Voelker, and S. Savage, "Inferring Internet denial-of-service activity," in *Proceedings of the 10th USENIX Security Symposium*, Washington, D.C., Aug. 2001.

[6] N. Brownlee, K. C. Claffy, and E. Nemeth, "DNS measurements at a root server," in *Proceedings of the IEEE GlobeCom*, San Antonio, USA, Nov. 2001.

[7] P. Albitz and C. Liu, *DNS and BIND*, 4th ed. Sebastopol, California, USA: O'Reilly & Associates, Inc., Apr. 2001.

[8] Men&Mice, "Domain health survey for .COM," Feb. 2003. [Online]. Available: http://www.menandmice.com/dnsplace/healthsurvey.html

[9] P. Vixie, G. Sneeringer, and M. Schleifer, "Events of 21-Oct-2002," ISC/UMD/Cogent, Tech. Rep., Nov. 2002.

[10] R. Bush, D. Karrenberg, M. Kosters, and R. Plzak, "Root name server operational requirements," Internet Engineering Task Force, Request for Comments RFC 2870, June 2000.

[11] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS performance and the effectiveness of caching," *IEEE/ACM Trans. Networking*, vol. 10, no. 5, pp. 589–603, Oct. 2002.

[12] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of DNS-based server selection," in *Proceedings of the IEEE INFOCOM*, Anchorage, USA, Apr. 2001.

[13] P. Mockapetris, "Domain names - concepts and facilities," Internet Engineering Task Force, Request for Comments RFC 1034, Nov. 1987.

[14] ——, "Domain names - implementation and specification," Internet Engineering Task Force, Request for Comments RFC 1035, Nov. 1987.

[15] A. Householder, A. Manion, L. Pesante, G. M. Weaver, and R. Thomas, *Managing the Threat of Denial-of-Service Attacks*. CERT Coordination Center, Oct. 2001.

[16] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, USA, Mar. 1999.

[17] L. A. Adamic, "Zipf, Power-laws, and Pareto - a ranking tutorial," Xerox Palo Alto Research Center, Tech. Rep., 2000. [Online]. Available: http://ginger.hpl.hp.com/shl/papers/ranking

[18] A. Kumar, J. Postel, C. Neuman, P. Danzig, and S. Miller, "Common DNS implementation errors and suggested fixes," Internet Engineering Task Force, Request for Comments RFC 1536, Oct. 1993.

[19] N. Brownlee and I. Ziedins, "Response time distributions for global name servers," in *Proceedings of the Passive & Active Measurement Workshop*, Fort Collins, Colorado, USA, Mar. 2002.

[20] S. Weber and L. Cheng, "A survey of anycast in IPv6 networks," *IEEE Commun. Mag.*, vol. 42, no. 1, pp. 127–132, Jan. 2004.

A Distributed Defense Mechanism Against Low–Bandwidth TCP SYN Flooding Attacks

Emmanuel Guiton
Helsinki University of Technology
Networking Laboratory
P.O. Box 3000
FIN–02015 HUT, Finland
Email: emmanuel.guiton@hut.fi

***Abstract*— TCP servers have typically few resources dedicated to connection establishments; only a small amount of half–open connections can be queued. TCP SYN flooding attacks perform denial of service by exhausting this backlog buffer capacity. Typical local networks include normal workstations whose backlog queues are not used as they are not meant to be servers. Thus, these networks have a pool of available resources that can be used to release a server from a flood of TCP connection requests. This paper presents a new firewall–based defense mechanism that silently distributes the load of TCP SYN requests amongst several hosts when a server is overloaded. A prototype implementation showed the viability of this end–system defense approach as long as the bandwidth capacity of the network links is not the bottleneck resource. This method can be easily implemented and a prototype showed that 3 computers can withstand a 280 kb/s (650 messages/s) TCP SYN flooding attack.**

***Index Terms*— Denial of Service, TCP SYN flooding, firewall, Network Address Translation, Connection Request Distribution.**

I. Introduction

In spite of its oldness, the TCP SYN flooding technique [1] is still one of the most common methods to perform Denial of Service (DoS) attacks [2]. Well-known DoS attack tools such as TFN [3] [4], TFN2k [5], Stacheldraht [6], and also viruses or worms like W32/Blaster [7] include this functionality. Several mechanisms have been proposed to cope with that problem but no complete solution still exists. Some issues are directly related to the place of the defense mechanisms on the attack path: the closer to the attack sources they are, the more false–positives they trigger. In return, the closer to the target, the less able to cope with flooding they are. However, when bottleneck links are not overloaded, end–system defenses can be the most effective and reliable. [1] presents such methods, including firewall based proposals like the one described hereafter.

This paper proposes a new approach that aims to distribute the TCP connection establishment load amongst several hosts in a local network. The mechanism is managed by a firewall that forwards and redirects messages so that connection requests are always sent to an host which has available resources. First, every connection request is normally forwarded to the server. When an attack occurs and the server's backlog queue gets full, the firewall uses the Network Address Port Translation (NAPT) method to redirect the TCP SYN requests towards one or several another host in the local network, which we will call relays. When a connection is established with a relay, the firewall moves it to the real server so that the legitimate client and the server can communicate normally.

In the next section, overall information about DoS attacks, the TCP SYN flooding mechanism, and NAPT is provided. Then, two firewall–based defense mechanisms closely related to the proposal in this study are presented. Following is the description of the proposal itself with particular attention on its critical issues. This proposal has been implemented and tested in a Linux environment. The results are provided in the last but one section before concluding this paper.

II. Background

A. Denial of Service attacks

DoS attacks aim to deny legitimate users the access to a service. A famous example is the wave of attacks in the early February 2000, when the websites of Amazon, CNN, eBay, Yahoo!, etc. were unreachable during several hours [8]. More recently, the SCO Group, Inc. has suffered a distributed DoS attack against its website from 1st to 12th of February 2004, rendering it inaccessible. To maintain the website accessibility, the company chose to change the address and to remove the original one from the DNS databases.

DoS attacks can be separated into two different categories. Logic attacks rely on intelligent exploitations of software vulnerabilities. Flooding attacks aim to exhaust a resource such as link bandwidth, processing power, disk space, etc. This second category includes the TCP SYN flooding attack, described in detail in [1]. The TCP SYN flooding was the relying technique used in the attacks mentioned above. The W32/Novarg.A virus (also called Mydoom) [9], which was used in the attack against SCO illustrates that this method is still used and very effective. From now on, we will concentrate on that particular kind of attack.

B. Defense methods

To defend against TCP SYN flooding attacks, general methods against DoS attacks and also several specific methods

can be used. Specific methods aim to improve the attack resistance or the reliability of the TCP connection establishment mechanism. Such methods include an optimization of servers' resources, the use of firewalls [1], active monitors [1], puzzle auctions [10], and so on. Some other defense mechanisms like blocking or several systems based on rate–limiting [11] [12] [13], aim to disrupt the flooding itself and can generally be applied to mitigate other flooding attacks. Usually, these methods also rely on traffic identification and selection, for the purpose of which Intrusion Detection Systems (IDS) are often used. Defense techniques also differ by their places on the attack path. While the first quoted ones are implemented on the target side, anti–flooding methods gain in being applied as close as possible to the attack sources. For that reason, a lot of recent work has focused on mechanisms that are able to trace attacks back to their sources [12], [14], [15], [16], [17], [18], [19], [20], [21].

All of these proposals have their own advantages and drawbacks. The closer to the target, the better the reliability of attack detection usually is. Traceback and rate–limiting techniques are commonly based on statistical analyses, which are liable to false–positives. Thus, legitimate users can be denied services because of these methods. On the other hand, when the defense is implemented close to the target, it cannot defend against every DoS attack. It is particularly useless against any high–bandwidth flooding attack since it cannot avoid the overload in upstream links. However, if waste of resources is not considered, it is worth noticing that an attack should not be mitigated in the upstream nodes if the target can handle it without problem. Otherwise, the defense mechanisms would risk denying services to some legitimate users without compensating with positive effects. Therefore, reliable and effective end–target defense methods are still useful, or even necessary. Moreover, it has been shown in [2] that more than 90% of the attacks on the Internet during a three week period in the beginning of year 2001 were TCP based and most probably used the TCP SYN mechanism. We can further deduce from the results in [2] that around 80% of the attacks used less than 2000 packets per second. Considering a TCP message size of 54 bytes (a TCP SYN message witout options), a bottleneck link of 1Mb/s requires 2315 messages to be overwhelmed. Thus, a reliable end–target defense method can be expected to be more effective and trigger less undue damage to legitimate users in a good proportion of attacks. However this proportion cannot be reliably deduced from [2]. For example, attack bandwidths may be adapted to the capacities of the targets' connections to the Internet. Moreover, since 2001, the substructures of the Internet has continued to grow worldwide in size and also in bandwidth capacity. Although the data given in [2] may be outdated, we can only refer to them as more recent measurements on the Internet DoS activity have not been published.

C. Network Address Port Translation

The security mechanism described in this paper is meant to be used in a local network, protected by a firewall from the public network. Indeed, the method relies on the Network Address Translation (NAT) principle, whose first description was published in [22]. It enables assigning freely IP addresses inside a local network. When a host wants to communicate outside the local network, its address is mapped into another one, which is globally unique on the outer network (the Internet, for example). The Network Address Port Translation (NAPT) [23] extended the number of possible mappings by using the {IP address, TCP/UDP port} pair instead of the single IP address.

The application areas of NAT, NAPT, and the various variants on the method [24] overtook the original goals, which was to provide a short term solution to the shortage of IPv4 addresses. Particularly, applications in security and network administration were quickly understood [25].

III. Related Work

Firewall–based defenses against TCP SYN flooding have already been proposed. In addition to a precise analysis of the TCP SYN flooding attack, [1] proposes several defense methods against it, including two firewall–based approaches.

A. Firewall as a relay

The first of these two proposals uses a firewall as a relay. The firewall is located at the interface between the local and the public network. When a TCP SYN request from the public network is sent to a local host, the firewall answers on its behalf. If the final TCP ACK message of the three-way handshake never arrives, the firewall terminates the connection. On the contrary, if the three-way handshake proceeds correctly, the firewall creates a second connection between itself and the true destination so that the communication path is completed. After that, the firewall acts as a proxy, forwarding the datagrams and translating the sequence numbers between the two ends of the connection. This method avoids that the destination host receives any spoofed TCP message. However, as reported in the same paper [1], this mechanism works only if the firewall is not itself vulnerable to TCP SYN flooding attacks. Indeed, the resource allocation issue is simply moved from the server to the firewall. Moreover, this method adds delay to every legitimate connection. Finally, the timeout period must be carefully selected; otherwise legitimate hosts with long response times may not be able to establish TCP connections.

B. Firewall as a semi–transparent gateway

The second firewall–based proposal in [1] uses a firewall as a semi–transparent gateway. In this case, the firewall monitors and reacts to the traffic exchanged between a local network and a public network. Unlike in the previous method, the TCP SYN and TCP ACK messages go through the firewall. However, when an internal host generates a TCP SYN+ACK message, the firewall establishes the connection with the local host by sending a TCP ACK message to it. This operation aims to free the resources allocated for the half–open connection in the local host's backlog queue. Once in this situation, two cases can occur. Firstly, the TCP SYN message can be part

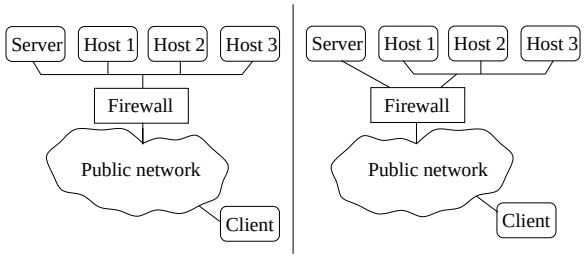


Fig. 1. Typical network environments of the proposal.

of an attack. Then, no TCP ACK message is generated by the source host and after a defined length of time the firewall sends a TCP RST message to terminate the connection at the local host side. Secondly, the TCP SYN message can be issued by a legitimate station, which finishes the three-way handshake by sending a TCP ACK message. This second TCP ACK message reaches normally the internal host and it does not create any problem since the TCP protocol is designed to cope with duplicate messages. Thus, the second TCP ACK message is silently discarded and the connection can continue without any additional operations from the firewall. As also explained in [1], this method does not create additional delays. However it transforms half–open connections into illegitimate open connections. As the resource limit on open connections (mainly processing power and memory) is much higher than the resource limit on half–open connections, this mechanism is both the main advantage and the main drawback of the proposal. Also, for the same reasons than in the previous method, the timeout length has to be carefully selected.

IV. Firewall as a Connection Request Distributor

A. Description of the defense mechanism

The goal of a TCP SYN flooding attack is to fill the victim's backlog queue with pending half–open connection. In many real-life environments, the target is typically part of a local network or it can be placed in a different but adjacent local network. These cases, represented in Fig. 1, are commonly found in small and medium–size companies. Most of the hosts in such local networks are normal workstations which are used, for example, for office work. Thus, they seldom use their resources to accept TCP connections as they are not configured to provide TCP services. These workstations form a pool of available resources that can be used to release an overloaded server of TCP connection establishments. In that way, there can most probably be enough resources to handle any flood of TCP SYN requests as long as network link capacity is not the bottleneck resource.

The defense mechanism relies on using a firewall that performs NAPT [23]. The transport layer information must be accessible as actions are taken according to the establishment status of connections. When a client wishes to connect to the server from the public network, it uses the IP address of the firewall or directly the address of the website if its publicly available (when it is separated from the rest of the network

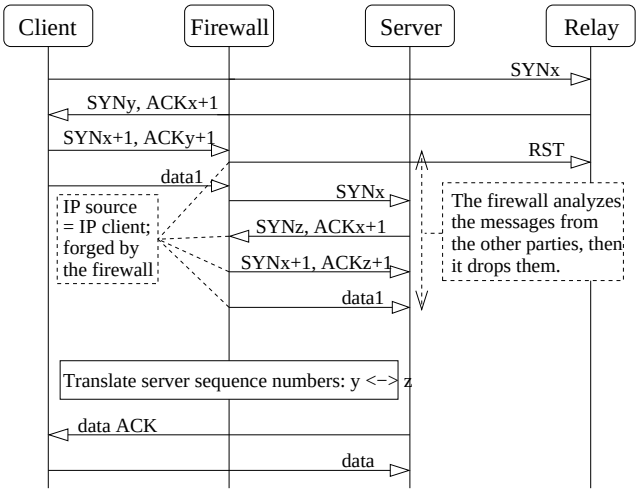


Fig. 2. Sequence chart of a legitimate client connection when redirected to a relay host.

like the case on the right in Fig. 1). None of the hosts involved in the process (server, local hosts, remote client) need to be aware of the presence of the firewall. Thus, the actions of the firewall are transparent for all of the parties. At this point, two cases are possible:

- The server is not under attack.
  The firewall directs the client's messages to the server, possibly performing usual NAPT operations, which improve the security of the local networks (by hiding local addresses to the hosts in the public network).
- The server is under attack and its queue of half–open connections is nearly full.
  The firewall fairly redirects the TCP SYN requests on a few hosts in the local network, which we will call relays. Relays are only used to establish TCP connections and every successful connection is moved back to the server. The relays only need an available port on which the TCP three–way handshakes can be performed; it does not even need to accept any data. The proceeding of the operations is depicted in the sequence chart of the Fig. 2. The firewall is used as a man–in–the–middle to act in behalf of the client. For that purpose, it must record some client data when tracking the connection so that it is able to forge messages to establish a connection with the real server. The firewall does not have direct knowledge of the relay's resource statuses, it only deduces them thanks to pre–configured parameters and calculations.

B. Critiques

This proposal addresses the drawbacks of the two firewall-based systems proposed in [1]. Here, the firewall is not vulnerable to TCP SYN flooding as it does not manage the connections itself but it only forwards the packets. The end hosts of a TCP connection need to allocate several data structures such as the socket structure and the TCP and IP control block structures in BSD style network code [1].

These structures include data and buffers which are used for sending and receiving; operations which are not performed by the firewall in the connection request distribution (CRD) scheme. In that case, the firewall uses only a limited amount of resources; mainly to store the critical data it needs to monitor connections. The firewall will only allocate the data structures used in TCP connections when forging packets. This case only happens when a legitimate connection is established with a relay. Four messages are forged (see Fig. 2), then the data structures are released.

Moreover, as long as the server is not under attack, legitimate connections do not suffer from additional delay. Extra-delay only occurs during the connection establishment when an attack is in progress. Compared to the second proposal in [1], there is no illegitimate connections that are created. Finally, in the firewall as a CRD system, there is more liberty in choosing the TCP timeout period as the mechanism does not only rely on the resources of one but of several stations.

Compared to other defense mechanisms like the ones referenced in the second section, this defense system ensures that none of the legitimate connection requests is dropped. Moreover, using NAPT enables hiding the infrastructure of the local network. Thus, there is no risk that an attacker learns about the local hosts, they cannot be distinguished from the server. The firewall as a CRD system is also a transparent mechanism for all clients in the public network and servers and hosts in the local network. Thus, this proposal can be easily implemented; it only requires modifying the firewall software. As only the TCP three-way handshake is allowed with the local hosts, the CRD system cannot be abused to attack them, except when exploiting a faulty implementation of the TCP protocol.

This defense mechanism has also its own drawbacks. It does not avoid the illegal consumptions of certain resources: link bandwidth, TCP ports as NATP identifies the different connections thanks to the {IP address, TCP port number} pairs and memory to store the data required to track connections. However, the availability of these resources exceed the consumption capabilities of the attacker inside the local network. In the case of a successful connection establishment, the systems generates 4 additional messages compared to a direct connection with the server. However this network load is negligible compared to the load triggered by the flooding attack. Finally, this solution relies on several programs that are not free from bugs and security vulnerabilities (e.g. [26]).

C. Critical issues

1) Counting the number of half–open connections: In order to control the distribution of the TCP connection requests, the firewall has to know the state of the half–open queues in the server and in the relays. These calculations are critical because the system can fail to detect the TCP request flooding if deduction is not properly done or if an attacker can deceive it. However, there is no need to involve the local hosts in the process as long as the firewall knows the TCP timeout length of the stations (but no synchronization is needed) and the maximum number of half–open connection they can handle. Then, the amount of pending half–connection can be calculated considering that:

- Each new TCP connection attempt is considered as being one more half–open connection.
- Each TCP connection reaching the established state (a correct TCP SYN ACK message has been received) decreases the number of half–open connections by one.
- Each connection expiring without having reached the established state decreases the number of half–open connections by one.

2) Translating the Server's Send Sequence numbers: When establishing a connection with a relay, the client receives from it a certain Initial Send Sequence (ISS) number value. This value is critical for both parties to communicate correctly. However, when the connection is moved to the server, a new ISS number is generated (and blocked at the firewall, see Fig. 2). The server expects the client to use this new number while the client expects the server to use the ISS it received. In order to reconcile these two requirements, the firewall must calculate the gap between the two sequence numbers and then it has to keep on translating the sequence number values between the two hosts. Note that the client's ISS does not need to be translated as the firewall can correctly replay it when establishing the connection to the server.

The same issue also applies to a few other options. The selective acknowledgement option is directly linked to sequence numbers and must be changed accordingly. The timestamp option works in a fashion similar to the sequence numbers and also requires translating the server's values.

3) Storing data: When tracking the connections, the firewall must be aware about the IP addresses and TCP ports involved. It requires also additional data as connections established with relays must be moved to the real server. To forge the necessary messages, it must know the three different ISSs, the TCP and IP flags (such as no fragmentation, explicit congestion notification) and the possible optional fields like TCP timestamps.

The firewall must also store any data packet coming from the client while the connection is not yet established with the server. This issue does not require much resources: only one packet may be sent along with the SYN ACK packet and the memory will be released quickly as the connection is legitimate.

V. Implementation and Network Environment

A. Implementing the defense mechanism

The test implementation was carried out using the code of the netfilter/iptables open source project [27]. It was implemented on a Linux kernel version 2.4.23, to which was added a patch provided on the netfilter/iptable project website. The netfilter firewall is divided into two parts: a user level application, which simply allows users to provide configuration data, and several kernel modules, which perform the firewall and NAPT operations. Additions to both of these

parts were necessary to implement the connection request distributor.

B. Configuring the Linux server and hosts

To implement the defense system, the first step is to optimize the resources of the server and hosts, so that they can handle the highest possible load of TCP SYN requests by themselves. Using a Linux OS, the behaviour of the TCP protocol can be customized by modifying the files located in the /proc/sys/net/ipv4/ and /proc/sys/net/core directories. (Old versions of the Linux kernel may require to change directly all or part of these variables in the kernel code sources.) Several guides, like [28] and [29], indicate how to set the values contained in these files. A simple way to make the changes is to use the sysctl utility. The following parameters are of particular interest:

The tcp_max_syn_backlog file indicates the maximum number of TCP SYN requests that can be stored in the backlog queue. The default value is dependent on the host's memory but it can be extended. However this value will be practically limited by the host's resource capacities.

The tcp_synack_retries contains a figure that corresponds to the maximum number of TCP SYN ACK messages that a destination host will send before closing the connection when the source host is not responding. With the default value, which is 5, the timeout occurs after 3 minutes. To cope with TCP SYN flooding attacks, it is recommended to leave only 1 retransmission so that the connection timeouts after 9 seconds without receiving an ACK message.

Finally, the memory allocated to TCP reception buffers can be increased by changing the default values in the following files: core/rmem_max, core/rmem_default, ipv4/tcp_rmem. The whole memory allocated to TCP can also be increased in ipv4/tcp_mem.

C. The test network environment

The performances of the whole CRD system depend on the performances of each station involved in the process. TCP performances depend on both software and hardware thus they vary from host to host. In that regard, the tests were affected by a low–performance environment. The prototype was tested on a network made up of Linux PCs with 2.4.20-2.4.23 kernels, 100 to 400 MHz processors, less than 100 MB of memory except one host which had 256 MB. Figure 3 shows the overall arrangement of the hosts. The network links are 100 Mb/s ethernet.

The SYN flooding attacks were carried out by a kernel-level program and targeted an HTTP server. The experimentations were successfully conducted using different port numbers on the relays (21 (FTP), 22 (SSH), 23 (telnet), 80 (HTTP)...) to ensure that the system was not dependent on a particular service. By experimenting, it was found that the server and one relay could handle up to 1537 SYN requests at the same time while the best relay could handle twice that amount.

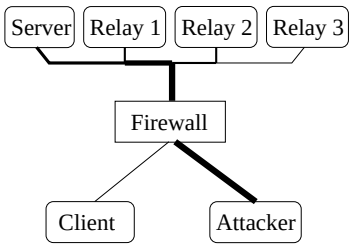


Fig. 3. The test network. The path of the TCP SYN flow is emphasized.

VI. Tests and Analyses

A. Overall performances of the CRD system

The main test was to determine the maximum attack bandwidth that the CRD system could handle. In order to find that limit, the flooding program was used with different sending rates during time intervals up to 10 minutes. In theory, the target and the two relays could handle up to 682 messages per second (almost 295kb/s). This rate is much less than what the network links and the network cards could stand. Thus, the TCP implementations on the different hosts, not the physical network, were the bottleneck. In practice, the experiments show that the CRD system could withstand a maximum of 650 messages per second (280kb/s) without a message being lost nor a legitimate connection failing.

The difference between practice and theory is due to the firewall's calculations. Regarding the target's and the relays' queue statuses, there was always a gap between reality and the estimations of the firewall. Minimizing this gap was one important issue to reach the 650 message per second limit.

It is worth noticing that the best relay was enduring half of the connection request load (140kb/s). 8 hosts like this one could withstand more than 1 Mb/s of TCP SYN flooding (1,12 Mb/s). Moreover, one should remember that the hardware equipments used for the tests do not reflect today's computers' capabilities. Using modern equipments can certainly lead to better results, but this could not be tested here.

B. Performances from the user point of view.

From the user point of view, this method produces nearly perfect results, only limited by the scope of the mechanism: low-bandwidth attacks. In case the server is not overwhelmed by connection requests, the user will connect directly to the server without performance degradation. On the other hand, while the server is flooded, a small delay happens: the length of time required to move the established connection from a relay to the server. Considering that the server and the relays are very close, probably on the same local network, this additional delay is very small when comparing to delays experienced in the public Internet, for example. In the tests described above, an average of 1.26 ms additional delay was measured for each connection which was first established with a relay while the system was under attack.

VII. Conclusion

This paper introduced a new end-system mechanism to defend a server against TCP SYN flooding attacks. The connection request distribution system uses a transport level packet based firewall to distribute the load of TCP SYN requests between several hosts. When a connection is established with a relay (the connection is legitimate), it is moved back to the real server. This method does not require special resources and it is also easy to implement (as an additional feature to existing firewall software), deploy and configure. A Linux–based prototype system showed the viability of the mechanism. Tests showed that a three-computer network could withstand 280 kb/s of TCP SYN flooding. The prototype should now being further developed and optimized to create a safely deployable system whose performances should easily approach theoretical bounds. The implementation flaws should be corrected and the design should be improved to cope with the complexity of real life environments (where potential relays are not always up and running, for example). This paper does not investigate all the performance aspects of the system. More tests could be performed with an improved version of the prototype and a network environment reflecting modern equipment capabilities. Particularly, performance issues in the firewall and the relays should be addressed. Finally, this paper introduced the idea of a distributed defense to address a particular issue: the TCP SYN flooding. The method may suit similar issues for other protocols.

References

[1] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on TCP," in *Proc. IEEE Symposium on Security and Privacy 1997*, May 4–7, 1997, pp. 208–223.

[2] D. Moore, G. M. Voelker, and S. Savage, "Inferring internet denial of service activity," in *Proc. Tenth USENIX Security Symposium*, Washington D.C., Aug. 2001.

[3] "Distributed denial of service tools," CERT Coordination Center, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, Incident Note IN–99–07, Jan. 15 2001. [Online]. Available: http://www.cert.org/incident_notes/IN–99–07.html#tfn

[4] D. Dittrich. (1999, Oct. 21) The DoS project's "trinoo" distributed denial of service attack tool. University of Washington. Seattle, Washington, USA. [Online]. Available: http://www.staff.washington.edu/dittrich/misc/trinoo.analysis

[5] "Denial of service tools," CERT Coordination Center, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, Advisory CA–1999–17, Mar. 3 2000. [Online]. Available: http://www.cert.org/advisories/CA–1999–17.html

[6] D. Dittrich. (1999, Dec. 31) The "stacheldraht" distributed denial of service attack tool. University of Washington. Seattle, Washington, USA. [Online]. Available: http://www.staff.washington.edu/dittrich/misc/stacheldraht.analysis

[7] "W32/blaster worm," CERT Coordination Center, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, Advisory CA–2003–20, Aug. 14 2003. [Online]. Available: http://www.cert.org/advisories/CA–2003–20.html

[8] L. Garber, "Denial–of–service attacks rip the internet," *IEEE Computer*, vol. 33, no. 4, pp. 12–17, Apr. 2000.

[9] "W32/novarg.a virus," CERT Coordination Center, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, Incident Note IN–2004–01, Jan. 30 2004. [Online]. Available: http://www.cert.org/incident_notes/IN–2004–01.html

[10] X. Wang and M. K. Reiter, "Defending against denial–of–service attacks with puzzle auctions," in *Proc. Symposium on Security and Privacy*, May 11–14 2003, pp. 78–92.

[11] A. Gaeil, K. Kiyoung, and J. Jongsoo, "MF (minority first) scheme for defeating distributed denial of service attacks," in *Proc. Eighth IEEE International Symposium on Computers and Communication (ISCC 2003)*, 2003, pp. 1233–1238.

[12] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 3, pp. 62–73, July 2002.

[13] D. Sterne, K. Djahandari, B. Wilson, B. Babson, D. Schnackenberg, H. Holliday, and T. Reid, "Autonomic response to distributed denial of service attacks," in *Proc. Fourth International Sympsium on Recent Advances in Intrusion Detection*, Davis, California, USA, Oct. 2001, pp. 134–149.

[14] S. Bellovin and T. Taylor. (2001, Oct.) ICMP traceback messages. Internet draft. [Online]. Available: www.globecom.net/ietf/draft/draft–ietf–itrace–01.html

[15] A. Mankin, D. Massey, W. Chien-Lung, S. F. Wu, and L. Zhang, "Autonomic response to distributed denial of service attacks," in *Proc. Tenth International Conference on Computer Communications and Networks*, Oct. 15–17 2001, pp. 159–165.

[16] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, "Hash–based ip traceback," in *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, San Diego, California, USA, 2001, pp. 3–14.

[17] W. T. Strayer, C. E. Jones, F. Tchakountio, A. C. Snoeren, B. Schwartz, R. C. Clements, M. Condell, and C. Partridge, "Traceback of single ip packets using spie," in *Proc. DARPA Information Survivability Conference and Exposition*, vol. 2, Davis, California, USA, Apr. 22–24 2003, pp. 266–270.

[18] H. Aljifri, "IP traceback: a new denial–of–service deterrent?" *IEEE Security & Privacy Magazine*, vol. 1, no. 3, pp. 24–31, May/June 2003.

[19] K. T. Law, J. C. S. Lui, and D. K. Y. Yau, "You can run, but you can't hide: an effective methodology to traceback ddos attackers," in *Proc. Tenth IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS 2002)*, Oct. 11–16 2002, pp. 433–440.

[20] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network support for ip traceback," *ACM IEEE Transactions on Networking*, vol. 9, no. 3, pp. 226–237, June 2001.

[21] M. T. Goodrich, "Efficient packet marking for large–scale ip traceback," in *Proc. Nineth ACM conference on Computer and communications security*, 2002, pp. 117–126.

[22] K. Egevang and P. Francis, "The ip network address translator (NAT)," RFC 1631, Internet Engineering Task Force, May 1994.

[23] P. Srisuresh and K. Egevang, "Traditional IP network address translator (traditional NAT)," RFC 3022, Internet Engineering Task Force, Jan. 2001.

[24] P. Srisuresh and M. Holdrege, "IP network address translator (NAT) terminology and considerations," RFC 2663, Internet Engineering Task Force, Aug. 1999.

[25] M. Smith and R. Hunt, "Network security using NAT and NAPT," in *Proc. 10th IEEE International Conference on Networks (ICON 2002)*, Aug. 27–30 2002, pp. 355–360.

[26] "Linux kernel netfilter irc dcc helper module creates overly permissive firewall rules," US-CERT, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, Vulnerability Note VU#230307, July 05 2002. [Online]. Available: http://www.kb.cert.org/vuls/id/230307

[27] (2004) The netfilter / iptables project homepage. [Online]. Available: http://www.netfilter.org/

[28] O. Andreasson. (2003, May 21) Ipsysctl tutorial 1.0.4. webpage. [Online]. Available: http://ipsysctl-tutorial.frozentux.net/ipsysctl-tutorial.html

[29] B. L. Tierney. (2001, Feb.) TCP tuning guide. webpage. [Online]. Available: http://www-didc.lbl.gov/TCP-tuning/TCP-tuning.html