FMS 2014

Formal Methods for Security

Proceedings of the Formal Methods for Security Workshop

A Satellite Event of Petri Nets 2014

Proceedings Editors:

Véronique Cortier, CNRS Researcher at LORIA, Lorraine University, France

Riadh Robbana, Professor in LIP2 & INSAT, University of Carthage, Tunisia

Tunis, Tunisia

June 23rd, 2014

Preface

This volume contains the proceedings of the 5th Workshop on Formal Methods for Security (FMS) held in Tunis, Tunisia, June 23, 2014 as satellite event of Petri Nets 2014.
FMS is dedicated to the advancement of the theory and practice of formal methods for security.
We received 9 submissions. These submissions went through a rigorous review process; each submission was reviewed by at least 3 Program Committee members. The Program Committee members have selected 5 papers for publication. In addition to the five selected papers, two more papers were selected for an informal presentation at the workshop.

We also have the chance to welcome one invited talk by Mark Ryan « Du-Vote: Remote Electronic Voting with Untrusted Computers » (joint work with Gurchetan Grewal, Michael Clarkson, Liqun Chen).

We hope you will enjoy the FMS 2014 edition!

Véronique Cortier and Riadh Robbana

Program Committee

Myrto Arapinis	University of Birmingham
Kamel Barkaoui	Cedric-Cnam
Narjes Ben Rajeb	LIP2 - INSAT - University of Carthage
Vincent Cheval	School of Computer Science, University of Birmingham
Stephen Chong	Harvard University
Veronique Cortier	CNRS, Loria
Stephanie Delaune	CNRS, LSV
Susanna Donatelli	Dipartimento di Informatica, Universita' di Torino
Sibylle Froeschle	University of Oldenburg
Pierre-Cyrille Heam	LSV, ENS Cachan, INRIA-CNRS
Béchir Ktari	Laval University
Yassine Lakhnech	VERIMAG
Mahjoub Langar	LIP2 - ENIT - Elmanar University
Mohamed Mejri	laval
Riadh Robbana	LIP2 - INSAT - University of Carthage
Hassen Saidi	SRI International
Lilia Sfaxi	INSAT - University of Carthage
Jacques Traore	Orange Labs

Additional Reviewers

Berrima, Mouhebeddine	University of Sousse
Yu, Jiangshan	University of Birmingham

Table of Contents

Du-Vote: Remote Electronic Voting with Untrusted Computers (Invited Talk)

Mark Ryan, Gurchetan Grewal, Michael Clarkson, and Liqun Chen

University of Birmingham, UK
m.d.ryan@cs.bham.ac.uk

Abstract. Du-Vote is a new remote electronic voting protocol that eliminates the often-required assumption that voters trust general-purpose computers. Trust is distributed in Du-Vote between a simple hardware token issued to the voter, the voters's computer, and a server run by election authorities. Verifiability is guaranteed with statistically high probability even if all these machines are controlled by the adversary, and privacy is guaranteed as long as at least either the voter's computer or the server is not controlled by the adversary. The design of the Du-Vote protocol is presented in this paper. A new non-interactive zero-knowledge proof is employed to verify the server's computations. The security of the protocol is analyzed to determine the extent to which, when components of the system are malicious, privacy and verifiability are maintained.

Keywords: du-vote, electronic voting protocol

Extraction of Insider Attack Scenarios from a Formal Information System Modeling

Amira Radhouani[1,2,3,5], Akram Idani[1,2], Yves Ledru[1,2], Narjes Ben Rajeb[3,4]

[1] Univ. of Grenoble Alpes, LIG, F-38000 Grenoble, France
[2] CNRS, LIG, F-38000 Grenoble, France
[3] LIP2-LR99ES18, 2092, Tunis, Tunisia
[4] INSAT - Carthage University, Tunisia
[5] FST - Tunis-El Manar University, Tunisia

Abstract. The early detection of potential threats during the modelling phase of a Secure Information System is required because it favours the design of a robust access control policy and the prevention of malicious behaviours during the system execution. This paper deals with internal attacks which can be made by people inside the organization. Such attacks are difficult to find because insiders have authorized system access and also may be familiar with system policies and procedures. We are interested in finding attacks which conform to the access control policy, but lead to unwanted states. These attacks are favoured by policies involving authorization constraints, which grant or deny access depending on the evolution of the functional Information System state. In this context, we propose to model functional requirements and their Role Based Access Control (RBAC) policies using B machines and then to formally reason on both models. In order to extract insider attack scenarios from these B specifications our approach first investigates symbolic behaviours. The use of a model-checking tool allows to exhibit, from a symbolic behaviour, an observable concrete sequence of operations that can be followed by an attacker. In this paper, we show how this combination of symbolic execution and model-checking allows to find out such insider attack scenarios.
keywords: Information System, B-Method, RBAC, attack scenario, Model Checking, Symbolic Search.

1 Introduction

Developing secure Information Systems remains an active research area addressing a wide range of challenges mostly interested in how to prevent from external attacks such as intrusion, code injection, denial of service, identity fraud, etc. Insider attacks are less addressed despite they may cause much more damage because an insider is over all a trusted entity. Intrinsically it is given means to violate a security policy, either by using legitimate access, or by obtaining unauthorized access. This paper deals especially with Role Based Access Control (RBAC) concerns with the aim to exhibit potential insider threats from a formal modelling of secure Information Systems. We are interested in finding

attacks which conform to the access control policy, but lead to unwanted states. These attacks are favoured by policies involving authorization constraints, which grant or deny access depending on the evolution of the functional Information System state. This reveals, on the one hand, the need to link the security model to the functional model of the information system, and on the other hand, to build tools taking into account the dynamic evolution of the IS state.

Tools such as SecureMova [2] and USE [6] are dedicated to validate security policies related to a functional model. But these tools don't take into account dynamic evolution of the functional state. In [10], we discussed shortcomings of existing approaches in this context, and showed the advantages of using a formal specification assisted by animation tools. This paper goes a step further than our previous works by taking advantage of model-checking and proof tools in order to automatically find insider attack scenarios composed of a sequence of actions modifying the functional state and breaking the authorization constraint.

This paper is organized as follows: section 2 gives an overview of our approach and its underlying methodology. In section 3 we present a simple example that illustrates our contribution. Section 4 defines semantics and technical aspects. In section 5 we propose a symbolic search that automates generation of attack scenarios and we discuss results of its application on the given example. Finally, we draw conclusions and perspectives.

2 Overall Approach

Bridging the gap between formal (*e.g.* Z, B, VDM ...) techniques and graphical languages such as UML has been a challenge since several years. On the one hand, formal techniques allow automatic reasoning assisted by proof and model-checking tools, and on the other hand, graphical techniques allow visualization and better understanding of the system structure. These complementary aspects are useful to ensure a software development process based on notations with precise syntax and semantics and which allows to structure a system graphically. Most existing research works [1, 5, 7, 13] in this context have been focused only on modelling and validation of functional aspects which are initially described by various kinds of UML diagrams (class, state/transition, sequence, ...) and then translated into a formal specification. These works have shown the interest of linking formal and graphical paradigms and also the feasibility of such translations.

In our work, we adopt a similar approach in order to graphically model and formally reason on both functional and security models. We developed the B4MSecure[6] platform [9] in order to translate a UML class diagram associated to a SecureUML model into B specifications. The resulting B specifications illustrated in figure 1 follow the separation of concerns principles in order to be able to validate both models separately and then validate their interactions.

The functional B model on the left hand side of figure 1 is issued from a conceptual class diagram. It integrates all basic operations generated automatically

[6] http://b4msecure.forge.imag.fr/

(constructors, destructors, setters, getters, ...) and also additional user-defined operations which are integrated into the graphical model and specified using the B syntax. This functional specification can be further improved by adding invariants and carrying out proof of correction with the help of AtelierB prover.

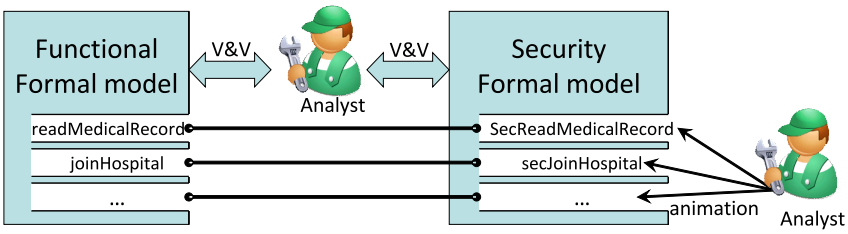


Fig. 1. Validation of functional and security models

The security model, on the right hand side of figure 1 is dedicated to control the access to functional operations with respect to access control rules defined in the SecureUML model. In our approach, we don't deal with administration operations because we make the simplifying assumption that access control rules don't evolve during the system execution. The security formal model allows to validate RBAC well-formedness rules such as no role hierarchy cycles, and separation of duty properties (SoD) such as assignment of conflicting roles to users...

This paper assumes that validation of both models in isolation is done: operations of functional model don't violate invariant properties, and the security model is robust. Such validation activities are widely discussed in the literature [12]. However, currently available validation approaches do not take sufficiently into account interactions between both models which result from the fact that constraints expressed in the security model also refer to information of the functional model. In fact, security policies often depend on dynamic properties based on the functional system state. For example, a bank customer may transfer funds from his account, but if the amount is greater than some limit the transfer must be approved by his account manager. Access control decisions depend then on the satisfaction of authorization constraints in the current system state. Dynamic evolution of the functional state impacts these constraints and may lead to a security vulnerability if it opens an unexpected access. In this paper we use validation tools (prover and model-checker) in order to search for malicious sequences of operations by analysing authorization constraints.

3 A Simple Example

In this section we use a running example issued from [2] and which deals with a SecureUML model associated to a functional UML class diagram.

3.1 Functional Model

The functional UML class diagram (presented in figure 2) describes a meeting scheduler dedicated to manage data about two entities: Persons and Meetings.

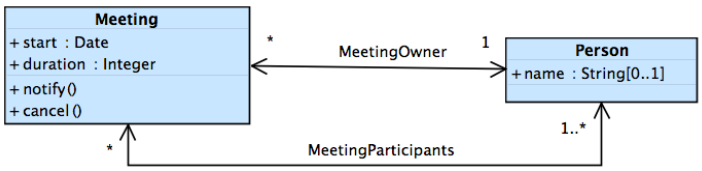


Fig. 2. Functional model of meeting scheduler system

A meeting has one and only one owner (association *MeetingOwner*), a list of participants (association *MeetingParticipants*), a duration, and a starting date. A person can be the owner of several meetings and may participate to several meetings. Operations *notify* and *cancel* are user-defined, and allow respectively to send messages to participants and to delete a meeting after notifying their participants by e-mail. Constructors, setters and getters are implicitly defined for both classes and both associations.

3.2 Access Control Rules

The access control model is given in Figure 3. It features three different roles:

- **SystemUser**: defines persons who are registered on the system and then have permission **UserMeetingPerm** which allow them to create and read meetings. Deletion and modification of meetings (including operation *cancel*) are granted to system users by means of permission **OwnerMeetingPerm**, featuring an authorization constraint checking that the user who tries to run these actions is the meeting owner.
- **Supervisor**: defines system users with more privileges because they can run actions notify and cancel on any meeting even if they are not owners.
- **SystemAdministrator**: having a full access on entity Person, an administrator manages system users. Full access grants him the right to create a new person, remove or modify an existing one. Furthermore, a system administrator has only a read access on meetings.

3.3 Validation

This example is intended to be validated in [2] based on a set of static queries that query a given system state in order to grasp some useful information like "*which user can perform an action on a concrete resource in a given state*".

Authorization constraint associated to **OwnerMeetingPerm** requires information from the functional model because it deals with the *MeetingOwner*

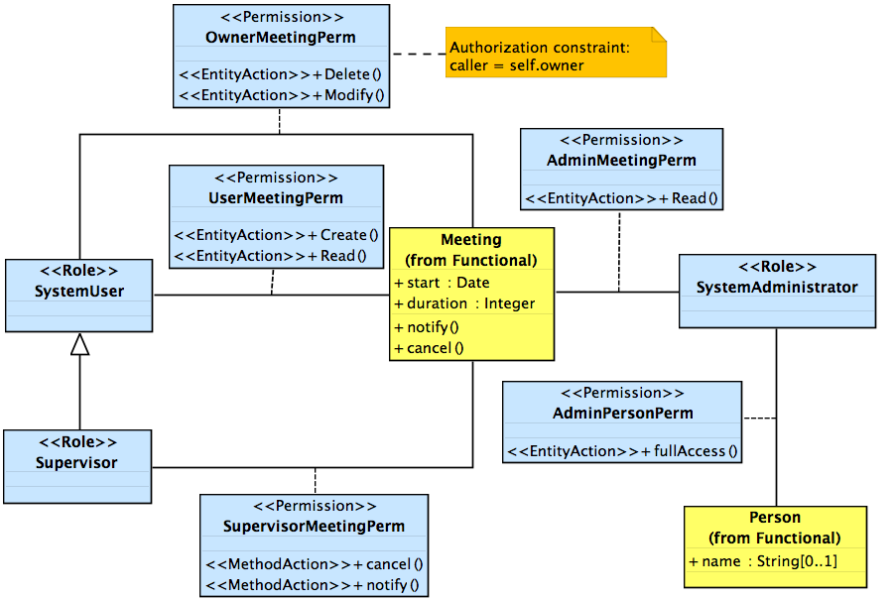


Fig. 3. Security model of meeting scheduler system

association. In the rest of this article, we consider three users John, Alice and Bob such that user assignments are as defined by figure 4 and a given initial state in which Alice is owner of meeting m_1, Bob is a participant of m_1. In such a state, the above static query establishes that only Alice is allowed to modify or delete m_1 because she is the owner of m_1.

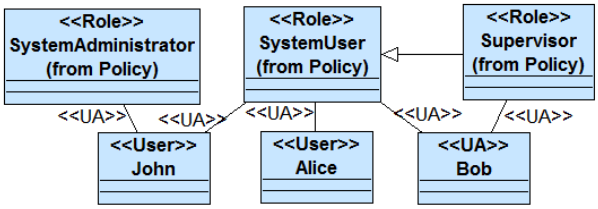


Fig. 4. Users assignement

In [10, 9] a dynamic analysis approach based on animation of a formal specification showed that validation should not only be based on a given static state, but should search for sequences of actions modifying this state and breaking the authorization constraint. For example, starting from the above state, a static query would only report that John, and also Bob, can't modify m_1 because none

of them satisfies the authorization constraint. A dynamic analysis would ask if there exists a sequence of operations enabled by John, or Bob, that allows them to modify m_1. This paper contributes towards automatically finding these malicious sequences. To perform these analysis, we applied the B4MSecure tool to the UML and SecureUML diagrams and generated a B specification counting 946 lines. This tool generates automatically a specification for all basic functional operations, which is enriched manually by some user-defined operations (*i.e.* cancel, notify).

4 Proposed Approach

4.1 Trace Semantics for B Specifications

In order to find malicious behaviours of an operational secure IS modelling, we rely on the set of finite observable traces of our B specifications. Indeed, B specifications can be approached by means of a trace semantics composed of an initialization substitution *init*, a set of operations $\mathcal{O}$ and a set of state variables $\mathcal{V}$. We note *val* a possible state predicate allowed by the invariant and *op* an operation from $\mathcal{O}$. A functional behaviour is an observable sequence $\mathcal{Q}$

$$\mathcal{Q} \mathrel{\hat{=}} init \ ; \ op_1 \ ; \ op_2 \ ; \ \ldots \ ; \ op_m$$

such that $\forall i.(i \in 1..m \Rightarrow op_i \in \mathcal{O})$ and there exists a sequence $\mathcal{S}$ of state predicates which does not violate invariant properties:

$$\mathcal{S} \mathrel{\hat{=}} val_0 \ ; \ val_1 \ ; \ \ldots \ ; \ val_m$$

in which val_0 is an initial state, and op_i is enabled from state val_{i-1} and state val_i is reached by op_i, starting from state val_{i-1}.

The security model filters functional behaviours by analysing access control premises which are triplets (u, R, c) where u is a user, R is a set of possible roles assigned to u, and c is an authorization constraint. An observable secure behaviour is a sequence $\mathcal{Q}$, where for every step i, premise (u_i, R_i, c_i) is valid (expressed as $(u_i, R_i, c_i) \models true$). This means that roles R_i activated by user u_i grant him the right of running operation op_i and if a constraint c_i exists, then it must be satisfied. The following premises sequence $\mathcal{P}$ must be valid for $\mathcal{Q}$:

$$\mathcal{P} \mathrel{\hat{=}} (u_1, R_1, c_1) \ ; \ (u_2, R_2, c_2) \ ; \ \ldots \ ; \ (u_m, R_m, c_m)$$

4.2 Tools to Exhibit Behaviours from B Specifications

Model-checking and symbolic proof techniques are of interest in order to exhibit a relevant behaviour from an operational B specification. Proof techniques deal with infinite systems and can prove constraint satisfiability, or establish that some operation can be enabled from an abstract state predicate. Model-checking is based on model exploration of finite systems, and can be used to find a sequence of actions leading to a given state or property. In our approach, we combine both techniques in order to overcome their shortcomings: complexity of proofs for the first one, and state explosion for the second one. In this sub-section, we illustrate both tools.

Model checking and animation (the ProB tool). ProB [11] is an animation and a model-checker of B specifications that explores the concrete state space of the specification and generates accessibility graphs. Then, every predicate val_i (where $i \in \{0, 1, \ldots, m\}$) of sequence $\mathcal{S}$ is a valuation of variables issued from $\mathcal{V}$. For example, starting from an initial state val_0 where:
$\mathcal{V} = \{person, meeting, meetingOwner, meetingParticipants\}$ and such that:

$$\begin{aligned} val_0 \mathrel{\hat{=}} & person = \emptyset \\ & \wedge meeting = \emptyset \\ & \wedge meetingOwner = \emptyset \\ & \wedge meetingParticipant = \emptyset \end{aligned}$$

and having $\mathcal{O} = \{personNew, meetingNew, meetingAddParticipants, \ldots\}$, the scenario of table 1 is successfully animated using ProB tool. Column "reached states" gives only modified B variables from the previous step.

step	Sequence $\mathcal{Q}$	Reached states $\mathcal{S}$	RBAC premises $\mathcal{P}$
1	personNew	person={Alice}	John SystemAdministrator no constraint
2	personNew	person={Alice, Bob}	John SystemAdministrator no constraint
3	meetingNew	meeting={m_1} meetingOwner={(Alice, m_1)}	Alice SystemUser no constraint
4	meetingAddParticipants	meetingParticipants={(m_1, Bob)}	Alice SystemUser Constraint: Alice is the owner of m_1

Table 1. animation of a normal scenario with ProB

In step 1, the tool animates operation $personNew$ which modifies variable person (initially equal to emptyset) and this action was performed by user John using role SystemAdministrator without need of authorization constraint. In step 4, the tool adds participant Bob to the meeting m_1 by animating operation meetingAddParticipants, after validating that authorization constraint is valid for Alice using role SystemUser. Indeed, Alice is the owner of m_1.

Symbolic proof (the GeneSyst tool). ProB is useful to animate scenarios identified during requirements analysis, or to exhaustively explore a finite subset of state space. As we are interested in finding malicious scenarios that exhibit a potential internal attack, the ProB technique may be useful only if it explores the right subset of state space in the right direction, which is not obvious for infinite systems. Symbolic proof techniques, such as that of GeneSyst [4], are more interesting because they allow to produce symbolic transition systems that represent a potentially infinite set of values. Such tools reason on the reachability

properties of a symbolic state F by some operation *op* from a symbolic state E. In [4], three reachability properties are defined in terms of the following proof obligations, where E and F are two disjoint state predicates:

(1) possibly reached: $E \wedge Pre(op) \Rightarrow \neg[Action(op)]\neg F$
(2) not reachable: $E \wedge Pre(op) \Rightarrow [Action(op)]\neg F$
(3) always reached: $E \wedge Pre(op) \Rightarrow [Action(op)]F$

In the generalized substitution theory, formula $[S]R$ means that substitution S always establishes predicate R, and $\neg[S]\neg R$ means that substitution S may establish predicate R. Hence, proof (1) means that state F can be reached by actions of operation *op*, when operation precondition is true in state E. Proof (2) means that F is never reached by actions of operation *op* from state E. Finally, proof (3) means that F is always reached by *op* from E. Let us consider, for example, the functional operation $meetingNew$:

$$
\begin{array}{l}
meetingNew(m,p) \hat{=} \\
\qquad\qquad PRE\ m \notin meeting \wedge p \in person\ THEN \\
\qquad\qquad\quad meeting := meeting \cup \{m\} \\
\qquad\qquad\quad || \ meetingOwner := meetingOwner \cup \{(m \mapsto p)\} \\
\qquad\qquad END
\end{array}
$$

This operation adds a new meeting m and links it to an owner p. If we define states E and F such that:

$$
\begin{array}{l}
E \ \hat{=}\ meetingOwner[\{m_1\}] = \emptyset \\
F \ \hat{=}\ meetingOwner[\{m_1\}] \neq \emptyset
\end{array}
$$

then proof obligation produced by GeneSyst for property (1) was successfully proved showing that operation $meetingNew$ when enabled from a state where m_1 does not exist and there exists at least one person in the system, may lead to a state where m_1 is created and has an owner.

Our work will be focused on proof (1) which states the reachability of a target state from an initial one, illustrated above, because it is sufficient to decide wether an operation is potentially useful for a malicious behaviour. Proofs (2) and (3) can be used if one would like to assume that a state can never be reached, or it is always reached, by an operation.

4.3 Malicious Behaviour

Based on the security requirements, several operations are identified as critical. For example, security requirements have identified the integrity of meeting information as critical. Therefore, operations which perform unauthorized modifications are identified as critical.

A malicious behaviour executed by a user u, regarding authorization constraints, is an observable secure behaviour $\mathcal{Q}$ with m steps such that:

- op_m is a critical operation to which an authorization constraint c_m is associated.
- user u is malicious and would like to run op_m by misusing his roles R_u.
- val_0 : is an initial state where $(u, R_u, c_m) \models false$
- for every step i ($i \in 1..m$) premise $(u, R_u, c_i) \models true$

In other words, malicious user u is not initially allowed to execute a critical operation, but he is able to run a sequence of operations leading to a state from which he can execute this operation. In our investigation we suppose that user u executes this malicious sequence without collusion. This problem will be tackled in a further work.

Section 3.3 gave an example where neither Bob nor John are allowed to run a modification operation, such as *meetingSetStart* which modifies attributes of class Meeting, from the initial state due to the authorization constraint. This initial state is:

$$\begin{aligned} val_0 \mathrel{\hat{=}} & person = \{Alice, Bob\} \\ & \wedge\ meeting = \{m_1\} \\ & \wedge\ meetingOwner = \{(Alice \mapsto m_1)\} \\ & \wedge\ meetingParticipant = \{(m_1 \mapsto Bob)\} \end{aligned}$$

In the following, we denote as $init_0$ the sequence of operations leading to val_0 such as that presented in table 1. We used the model-checking facility of ProB in order to explore exhaustively the state space and automatically find a path starting from val_0 and leading to a state where operation *meetingSetStart* becomes permitted to John. We asked ProB to find a sequence where John becomes the owner of m_1:

$$meetingOwner(m_1) = John$$

After exploring more than 1000 states, ProB found a scenario in which John executes sequentially operations personNew, personAddMeetingOwner and meetingSetStart. Indeed, this dynamic analysis showed that John, as a system administrator, has a full access to entity Person. This permission allows him to create, modify, read and delete any instance of class Person. First, he creates an instance John of class Person that corresponds to him by running operation $personNew(John)$. Then he adds meeting m_1 to the set of meetings owned by John, by running operation $personAddMeetingOwner(John, m_1)$ which is a basic modification operation of class Person. These two actions allowed him to become the owner of m_1 and then he was able to modify the meeting of Alice.
Like all model-checking techniques, when ProB explores exhaustively the state space, it faces the combinatorial explosion problem which depends on the number of operations provided to the tool and the state space size. In order to address this problem, our approach proposes a symbolic search which finds a sequence of potentially useful operations on which the model-checker should be focused.

5 Symbolic Search

The proposed symbolic search is performed by an algorithm that looks for an observable sequence $\mathcal{Q} \hat{=} init_0 \ ; \ op_1 \ ; \ \ldots \ ; \ op_m$ executed by a user u, and where (u, R_u, c_m) is not valid for a critical operation op_m in the initial state val_0 but becomes valid for state val_{m-1} where op_m can be enabled. It is a backward search algorithm, starting from the goal state val_{m-1} from which the critical operation op_m can be enabled: $val_{m-1} \hat{=} c_m \wedge Pre(op_m)$; and working backwards until the initial state val_0 is encountered. The algorithm ends when sequence $\mathcal{Q}$ is found or when all operations are verified without encountering the initial state. We consider that val_0 is a completely valuated state such as that where Alice is the owner of m_1, and Bob is a participant to m_1. This prevents the initial state from being included in both states val_{m-1} and val_{m-2}, which would never verify the condition of the while loop. Note that each operation occurs at most once in a computed sequence, which ensures the termination of our algorithm.

1. $\mathcal{Q} \hat{=} op_m$;
2. $val_{m-1} \hat{=} c_m \wedge Pre(op_m)$;
3. $val_{m-2} \hat{=} \neg val_{m-1}$;
4. **while** $val_0 \not\Rightarrow val_{m-1}$ **do**
5. **choose any** $o_i \in \mathcal{O}$ **where**
6. $(u, R_u, c_i) \models true \wedge$
7. $val_{m-2} \wedge Pre(o_i) \Rightarrow \neg[Action(o_i)]\neg val_{m-1}$
8. **do**
9. $\mathcal{Q} \hat{=} o_i \ ; \ \mathcal{Q} \ ;$
10. $val_{m-1} \hat{=} val_{m-2} \wedge Pre(o_i)$;
11. $val_{m-2} \hat{=} val_{m-2} \wedge \neg Pre(o_i)$;
12. **else**
13. **raise exception:** No sequence found
14. **enddo**
15. **endwhile**
16. $\mathcal{Q} \hat{=} init \ ; \mathcal{Q} \ ;$

5.1 Step by Step Illustration

We take advantage of abstraction and step by step we refine the val_{m-2} symbolic state:

1. At the first step of the algorithm, the state space is represented by two symbolic states: the first one val_{m-1} includes all states where the authorization constraint c_m is true and which are enabling op_m, and the second one val_{m-2} is the negation of val_{m-1} which is then $\neg c_m \vee \neg Pre(op_m)$. As they are two disjoint state predicates, we conduct proof (1) in order to find an operation o_i that belongs to $\mathcal{O}$ and which possibly reaches the first state val_{m-1} from the second one val_{m-2} and such that premise (u, R_u, c_i) is valid. If o_i does not exist, then no sequence could be found for the expected attack and we

can try proof (2) for each operation attesting that all operations never reach val_{m-1} from val_{m-2}.

2. At the second step of the algorithm, if the proof (1) succeeds for some operation op_{m-1}, then it may exist an observable sequence leading to the critical operation where access control premise (u, R_u, c_m) is valid, and hence a potential symbolic attack scenario can be found. The algorithm looks inside state val_{m-2} in order to find out the previous operations that can be invoked in the attack scenario. State val_{m-2} is partitioned into two sub-states which are:

$$val_{m-2} \wedge Pre(op_{m-1}) \equiv \neg(c_m \wedge Pre(op_m)) \wedge Pre(op_{m-1})$$
$$val_{m-2} \wedge \neg Pre(op_{m-1}) \equiv \neg(c_m \wedge Pre(op_m)) \wedge \neg Pre(op_{m-1})$$

Then, we look for operations that reach the first sub-state from the second one.

3. The algorithm proceeds iteratively by partitioning the second state into two sub-states until it finds a state that includes the initial state. In the best case, our algorithm gives some symbolic attack scenario, which consists of sequence $(init_0 \ ; \ op_n \ ; \ op_{n+1} \ ; \ \ldots \ ; \ op_m)$ invoked by the same user u and where:

$$val_{n-1} \hat{=} \neg(c_m \wedge Pre(op_m)) \wedge \neg Pre(op_{m-1}) \wedge \neg Pre(op_{m-2}) \wedge \ldots \wedge Pre(op_n)$$

and such that $val_0 \Rightarrow val_{n-1} \wedge \forall i.(i \in (n..m) \Rightarrow (u, R_u, c_i) \models true)$

5.2 Application

We apply our algorithm to the meeting scheduler example starting from the following initial state val_0:

$$\begin{aligned} val_0 \hat{=}\ & person = \{Alice, Bob\} \\ & \wedge\ meeting = \{m_1\} \\ & \wedge\ meetingOwner = \{(Alice \mapsto m_1)\} \\ & \wedge\ meetingParticipant = \{(m_1 \mapsto Bob)\} \end{aligned}$$

In this state user John is not allowed to modify meeting m_1 because the authorization constraint allows modification only for the owner of m_1. A malicious scenario would lead to a state where John becomes able to execute a modification operation such as operation $meetingSetStart$ on meeting m_1. In this state we have to verify:

$$\begin{aligned} & Pre(meetingSetStart(m_1, start)) \hat{=}\ m_1 \in meeting \wedge start \in NAT \\ & \text{and } (John, SystemUser, MeetingOwner(m_1) = John) \models true \end{aligned}$$

1. First iteration: considering the following symbolic states

$$val_{m-1} = (MeetingOwner(m_1) = John) \wedge m_1 \in Meeting \wedge start \in NAT$$
$$val_{m-2} = \neg val_{m-1}$$

we have:
– $val_0 \not\Rightarrow val_{m-1}$ because, in state val_0, $MeetingOwner(m_1) = Alice$, and
– proof (1) succeeds for the operations meetingNew and personAddMeetingOwner.
Then, we may go on the second iteration of the algorithm for each of these operations.

2. Second iteration: we partition state val_{m-2} into two sub-states:

$$val_{m-2} = \neg val_{m-1} \wedge Pre(op_{m-1})$$
$$val_{m-3} = \neg val_{m-1} \wedge \neg Pre(op_{m-1})$$

– **case 1:** we choose $op_{m-1} = meetingNew$, and then we have:
– $Pre(meetingNew) \mathrel{\hat{=}} m_1 \in meeting \wedge John \in person$, and
– $val_0 \not\Rightarrow val_{m-2}$ because $John \notin person$
In this case, the algorithm does not find an operation leading to a state where operation $meetingNew$ becomes enabled. Indeed, no operation satisfied proof obligation (1). Our algorithm concludes that it does not exist an attack scenario invoking $meetingNew$ at this step.

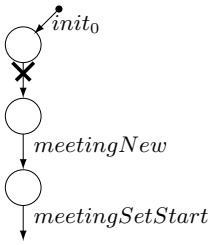


Fig. 5. No state enables operation $meetingNew$ is found

– **case 2:** we choose $op_{m-1} = personAddMeetingOwner$ and then we have:
– $Pre(personAddMeetingOwner) =$
$m_1 \in meeting \wedge John \in person \wedge (John, m_1) \notin MeetingOwner$
– $val_0 \not\Rightarrow val_{m-2}$ because $John \notin person$
In this case, proof (1) succeeds for operation $personNew$ which means that if $personNew$ is executed, it may lead to a state where $meetingNew$ can be enabled.

3. Third iteration: we partition state val_{m-3} into two sub-states:

$$val_{m-3} = \neg val_{m-1} \wedge \neg Pre(personAddMeetingOwner) \wedge Pre(personNew)$$
$$val_{m-4} = \neg val_{m-1} \wedge \neg Pre(personAddMeetingOwner) \wedge \neg Pre(personNew)$$

This stops normally the algorithm because in this case $val_0 \Rightarrow val_{m-3}$. Indeed, $Pre(personNew) \mathrel{\hat{=}} John \notin person$ and in the initial state John does not belong to set $person$. Figure 6 presents the full symbolic scenario that allows John to modify Alice's meeting.

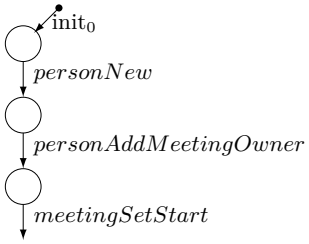


Fig. 6. Symbolic malicious behaviour for user John.

5.3 Discussion

Technically, our approach applies the GeneSyst tool in order to produce proof obligations and then asks the AtelierB prover to discharge them automatically. As the resulting scenarios are symbolic and based on "possibly reached proofs", the analyst can conclude that attacks may exist but he can not attest their feasibility for the concrete system. An interesting contribution of our proof-based symbolic sequences, besides the fact that they draw the analyst's attention to potential flaws, is that they give useful inputs to the model-checker. Indeed, a model-checking tool can be used to exhibit, from a symbolic behaviour, an observable concrete sequence of operations that can be followed by an attacker. In order to reduce significantly the state space, we can ask ProB to explore only operations found in the symbolic malicious scenarios. For our example, when trying only operations personNew, personAddMeetingOwner and meetingSetStart, ProB exhibits a concrete attack scenario after visiting a dozen of states which shows a significant speed up with respect to our initial ProB attempts (involving more than 1000 states).

Our technique was able to extract another scenario (figure 7) which can be executed by user Bob from the same initial state, in order to steal the ownership of m_1. In this scenario, Bob first cancels the meeting and then he recreates it before applying the critical operation.

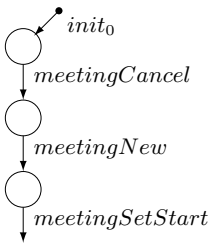


Fig. 7. Symbolic malicious behaviour for user Bob.

The first scenario, done by user John, is made possible by the full access permission to class Person, associated to role SystemAdministrator, which includes the right to modify association ends. This attack affects meeting integrity. One

solution can be to add a SSD constraint between roles SystemAdministrator and SystemUser. John will then still be able to become owner of the meeting, but will not be able to log in as SystemUser in order to modify it.

The second scenario done by Bob was possible due to role Supervisor which gives him the right to cancel a meeting, and then, as a SystemUser he can recreate it in order to become its owner. This scenario does not point out a flaw since whenever a meeting is cancelled it should be legitimate that a user can start a new meeting with the same identifier as the cancelled one.

6 Conclusion

We described in this paper a symbolic search approach that can extract insider malicious behaviours from a formal Information System modelling. The meeting scheduler example was discussed in several articles [3, 2]. However, they do not report the attack scenarios presented in this paper. This is due to the fact that dynamic evolution of the functional state is not taken into account. Contributions of this paper showed how dynamic analysis, assisted by proofs and model-checking, is useful to find out potential threats. In addition, thanks to our algorithm, proofs and model checking tools, our method can be fully automated in order to extract attack scenarios breaking authorization constraint. In [10], a dynamic analysis is done interactively with the help of a Z animator, but it is tedious and may miss many possible flaws. We also applied our approach on the case study that has been treated in [8] and we were able to find, automatically, the discussed threat. Currently we are looking for application on a real case study, issued from the ANR-Selkis project[7], and which deals with a medical information system involving various authorization constraints.

Our approach is automated by exploiting tools B4MSecure[8], GeneSyst[9], AtelierB[10] and ProB[11]. B4MSecure translates functional and security graphical models into B specification, from which we automatically produce proof obligations on reachability properties by taking advantage of the GeneSyst tool. Then, these proof obligations are discharged automatically using the AtelierB prover. When a symbolic scenario is found, ProB is used to explore concrete state space focusing on operations issued from the symbolic scenario. The main limitation of our work is that sometimes, when proof obligations are complex, AtelierB fails to prove them automatically. Interactive proofs are then required, but they may be pretty difficult for the analyst. One naive solution is to keep operations for which proofs don't succeed automatically in order to be exploited further using the model-checker. A more interesting solution is to focus on other kinds of proof obligations. For example, one can try to prove that an operation o is never enabled from a state E and/or o never reaches a state F. Applying

[7] http://lacl.univ-paris12.fr/selkis
[8] http://b4msecure.forge.imag.fr
[9] http://perso.citi.insa-lyon.fr/nstouls/?ZoomSur=GeneSyst
[10] http://www.atelierb.eu/
[11] http://www.stups.uni-duesseldorf.de/ProB

these proofs to the meeting scheduler example we were able to eliminate half of the operations after proving automatically that they cannot be involved in the attack scenario.

We believe that reachability properties can be expressed by means of LTL formula. We started exploring this direction basing on the LTL formula checking facilities of ProB and the first results are promising.

References

1. K. Anastasakis, B. Bordbar, G. Georg, and I. Ray. Uml2alloy: A challenging model transformation. *Model Driven Engineering Languages and Systems*, 2007.
2. D. Basin, M. Clavel, J. Doser, and M. Egea. Automated analysis of security-design models. *Information Software Technology*, 51, 2009.
3. D. Basin, J. Doser, and T. Lodderstedt. Model driven security: From uml models to access control infrastructures. *ACM Trans. Softw. Eng. Methodol.*, 15(1), 2006.
4. M-L. Potet D. Bert and N. Stouls. GeneSyst: a Tool to Reason about Behavioral Aspects of B Event Specifications. Application to Security Properties. In H. Treharne, S. King, M. C. Henson, and S. A. Schneider, editors, *ZB 2005: Formal Specification and Development in Z and B, 4th International Conference of B and Z Users*, volume 3455 of *LNCS*. Springer-Verlag, 2005.
5. A. Idani, J. Boulanger, and L. Philippe. Linking paradigms in safety critical systems. *International Journal of Computers and their Applications (IJCA), Special Issue on the Application of Computer Technology to Public Safety and Law Enforcement*, 16(2), 2009.
6. M. Kuhlmann, K. Sohr, and M. Gogolla. Employing uml and ocl for designing and analysing role-based access control. *Mathematical Structures in Computer Science*, 23(4), 2013.
7. K. Lano, D. Clark, and K. Androutsopoulos. UML to B: Formal Verification of Object-Oriented Models. In *Integrated Formal Methods*, volume 2999 of *LNCS*. Springer, 2004.
8. Y. Ledru, A. Idani, J. Milhau, N. Qamar, R. Laleau, J-L. Richier, and M-A. Labiadh. Taking into account functional models in the validation of is security policies. In Camille Salinesi and Oscar Pastor, editors, *CAiSE Workshops*, volume 83 of *Lecture Notes in Business Information Processing*. Springer, 2011.
9. Y. Ledru, A. Idani, J. Milhau, N. Qamar, Régine Laleau, J-L. Richier, and M-A. Labiadh. Validation of IS security policies featuring authorisation constraints. *International Journal of Information System Modeling and Design (IJISMD)*, 2014.
10. Y. Ledru, N. Qamar, A. Idani, J-L. Richier, and M-A. Labiadh. Validation of security policies by the animation of z specifications. In *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies*, SACMAT '11, New York, NY, USA, 2011. ACM.
11. M. Leuschel and M. Butler. ProB: A Model Checker for B. In *FME 2003: Formal Methods Europe*, volume 2805 of *LNCS*. Springer-Verlag, 2003.
12. N. Qamar, Y. Ledru, and A. Idani. Evaluating RBAC Supported Techniques and their Validation and Verification. In *6th International Conference on Availability, Reliability and Security (ARES 2011)*, Vienna, Autriche, August 2011. IEEE Computer Society.
13. C. Snook and M. Butler. UML-B: Formal modeling and design aided by UML. *ACM Transactions on Software Engineering and Methodology*, 15(1), 2006.

Traffic Analysis of Web Browsers

Sami Zhioua* and Mahjoub Langar**

*Information and Computer Sciences Department, KFUPM
P.O. Box 958, Dhahran 31261, KSA
zhioua@kfupm.edu.sa
**Ecole Nationale des Ingenieurs de Tunis
B.P. 37, Belveder, Tunis
mahjoub.langar@gmail.com

Abstract. Tor network is currently the most commonly used anonymity system with more than 300,000 users and almost 3000 relays. Attacks against Tor are typically confirmation attacks where the adversary injects easily discernible traffic pattern and observes which clients and/or relays exhibit such patterns. The main limitation of these attacks is that they require a "powerful" adversary. Website fingerprinting is a new breed of attacks that identifies which websites are visited by a Tor client by learning the traffic pattern for each suspected website. Recent works showed that some classifiers can successfully identify 80% of visited websites. In this paper we use a classic classifier, namely, decision trees (C4.5 algorithm) and we study to which extent popular web browsers can resist to website fingerprinting attacks. Among four studied web browsers, Google Chrome offers the best resistance to website fingerprinting (5 times better than the other web browsers). Since most of existing fingerprinting techniques have been evaluated using Firefox web browser, we expect the accuracy results of existing works to be reduced in case Chrome browser is used.

1 Introduction

Anonymity systems, such as Tor [1] and Jap [2] are designed primarily to provide privacy and anonymity to Internet users living in oppressive regimes giving them the opportunity to evade censorship. These systems achieve anonymity by embedding user data inside several layers of encryption and by forwarding the traffic through a set of relay nodes/proxies. This makes the job of an eavesdropping adversary much more challenging since by just observing the traffic she cannot deduce who is communicating with whom and what is the type of traffic exchanged.

Tor [1] represents the current state-of-the-art in low-latency anonymity systems. The Tor network is currently the largest deployed anonymity network ever, consisting of almost 3000 relays and more than an estimated 300,000 users.

Most of attacks against Tor anonymity system were traffic confirmation attacks where the idea was to inject easily discernable traffic pattern and observe which potential clients are exhibiting such patterns [3–7]. Most of these attacks

require a "powerful" adversary which is assumed to observe the traffic of a significant number of Tor relays and in some attacks to inject malicious relays in the Tor network. These assumptions are relatively strong and beyond the capabilities of most of attackers including totalitarian regimes. A more practical attack on Tor which does not require strong assumptions is passive *traffic analysis*. Traffic analysis consists in intercepting and analyzing the traffic messages (usually encrypted) in order to reveal information about the communication (e.g. the identities of the communicating entities, the type of data exchanged, etc.). To carry out the attack, the adversary is assumed to observe the traffic of only one side of the communication (usually the Tor client). This threat model is very common and holds particularly in presence of censorship.

Website fingerprinting [8] is a variant of passive traffic analysis that can be carried out by a local eavesdropper or by any entity observing Tor client traffic. In this attack the adversary analyzes the traffic to extract patterns that can reveal the identity of the website accessed by the client. Patterns are constructed from certain features in the traffic such as the size of transferred data, the timing, the order of packets, etc.

Website fingerprinting was first used to analyze encrypted HTTP traffic [8–11]. Most of these attacks were based on tracking the size the objects fetched by the main web page. With the migration to HTTP/1.1 which makes use of persistent connections and pipelining, it is no longer possible to easily distinguish between single objects fetching. Only few works focused on implementing website fingerprinting on anonymity systems [12–15]. It turned out that website fingerprinting is much challenging when applied on anonymity systems in particular Tor. The reason is that Tor protocol performs some structural modifications in the traffic: restructuring the traffic into fixed size cells, merging small packets together, multiplexing TCP streams, etc. However, despite these challenges, recent works showed that the precision of website fingerprinting could be as high as 80% when applied on Tor [15].

Tor protocol might be used with different web browsers[1]. Since web browsers use different user agents and process data packets differently, the choice of the web browser should have an impact on the efficiency of website fingerprinting. In this paper we study the impact of the choice the web browser on the anonymity of Tor clients with respect to website fingerprinting attacks. We consider a representative set of four popular web browsers, namely, Firefox, Chrome, Konqueror, and Internet Explorer, and we empirically analyze to which extent they resist to website fingerprinting. This is the first work in the literature that studies the efficiency of website fingerprinting while using different web browsers. All existing works were focusing on a single web browser, mainly Firefox.

The contributions of this paper are two-fold:

1. A detailed and complete survey of existing website fingerprinting approaches in particular targeting anonymity systems.
2. A comparative analysis of the most popular web browsers according to their resistance to website fingerprinting.

[1] Provided that the web browser allows to configure the socket proxy

2 Related Work

Early website fingerprinting techniques were focusing on analyzing simple encrypted HTTP traffic. Hintz [8], which is the first to use the term "fingerprinting" to refer to this type of attack, implemented a simple website fingerprinting attack targeting the SafeWeb encrypting web proxy [16]. The attack was based on tracking the size of objects fetched by a visited website. This was possible because the author did a strong assumption that every web object (image, ads, etc.) is fetched through a separate TCP connection using a different port. His experiment was a simple proof of concept distingushing only 5 websites. He achieved a detection rate between 45 and 75%.

Similarly, Sun et al. [9] based their approach on the size of fetched objects. Objects are isolated in the encrypted traffic by counting the number of packets between blocks of requests. The fingerprint is expressed as a multiset of object lengths. An unknown traffic sequence is then evaluated against website fingerprints using a measure of similarity (Jaccard's Similarity [17]). A similarity value more than a threshold c indicates a matching. In their empirical analysis, Sun et al. constructed a database of 2000 website fingerprints and then tried to distinguish these same 2000 websites out of a set of 100,000 websites. The optimal accuracy was obtained with the threshold c equal to 0.7 where 75% of the 2000 websites were correctly identified with a false positive rate of 1.5%.

The strong assumption that web objects can be distinguished by observing the different TCP connections does not hold anymore since with the migration to HTTP/1.1, no TCP connection is opened for each object as was the case in HTTP/1.0.

Bissias et al. [10] were the first to use IP packet sizes and Inter-Packet-Time (IPT) instead of the size of fetched objects to fingerprint websites. From every website visit they extract two traces: one size trace and one time trace. The size trace is the sequence of packet sizes while the time trace is the sequence of IPT times. Then all traces corresponding to a given website are merged into a website profile[2] by computing the arithmetic mean at every time step. Once a bank of profiles is constructed, an unknown network traffic is matched with each one of the constructed profiles using Cross Correlation [18]. The empirical analysis was based on the 100 websites most visited in the authors's department (University of Massachusetts) and showed that size profiles are much more efficient in identifying visited websites than time profiles. With size profiles, 20% of the analyzed traces are correctly identified after one guess and 65% after 10 guesses while with time profiles 8% of websites were correctly identified after one guess and 27% after 10 guesses. The analysis showed also that the time gap between the training phase (constructing the profiles) and the testing phase has only a small impact on the accuracy: a one hour gap is only 5% better than a 168 hours gap.

Liberatore et al. [11] obtained much better results by focusing only on packet sizes. In their work, they represented a traffic trace as a vector of packet size

[2] Actually two profiles: a size profile and a time profile

frequencies: each visit will result in a histogram of packet size frequencies. They tried two classification techniques: Jaccard's Similarity [17][3] and Naive Bayes [19]. The empirical analysis was also performed using the University of Massachusetts's typical traffic by filtering the top 2000 visited websites. Jaccard's based classification was slightly better than Naive based one with 73% of website visits correctly identified. Experiments showed also that the training set need not be very large since a training set of size 4 resulted in almost the same accuracy of training set of size 12.

All above works focused on website fingerprinting typical encrypted HTTP traffic. As anonymity systems became popular, recent website fingerprinting contributions focused on attacking those systems, in particular Tor [12–15].

Shi et al. [13] detailed a website fingerprinting attack on Tor. They adapted Hintz [8] and Sun et al. [9] techniques for Tor since instead of tracking the size of fetched objects (which is not possible in Tor), they tracked the number of packets sent or received in every interval[4]. A traffic trace is then represented by a vector specifying the number of intervals with 2 packets, the number of intervals with 3 packets, etc. Once a profile is built for a website (after several visits), the similarity between a profile and a traffic trace is computed using cosine similarity. The technique was evaluated using the traffic from the top 20 websites in Japan. They could identify successfully 50% of the visited websites.

Federrath et al. [12] used a Multinomial Naive Bayes (MNB) classifier to website fingerprint 6 anonymity systems: 4 single-hop proxy VPNs and 2 multi-hop systems: Tor and JonDonym [2]. As in Liberatore et al. [11] a traffic trace is represented as a histogram of packet size frequencies distribution without taking into consideration the packet ordering nor packet timing. They improved the efficiency of the MNB classifier by using text mining optimizations [19] such as Term Frequency Transformation, Inverse Document Frequency, etc. The evaluation was based on the top 2000 websites extracted from the log files of medium-range proxy server used by 50 schools. These 2000 websites has been filtered to 775 websites. The accuracy of their technique was very good for single-hop anonymity systems where 94% of website visits were correctly identified while it was relatively poor for multi-hop anonymity systems: 20% for JonDonym and only 3% for Tor. This shows once again that website fingerprinting is much more challenging with anonymity systems than with typical encrypted HTTP traffic.

Panchenko et al. [14] focused only on Tor and JonDonym and used SVMs (Support Vector Machines) for classification. They represented a traffic trace as a sequence of packet lengths where input and output packets are distinguished by using negative and positive values. In addition, they inject some features in these sequences to help in the classification such as size markers (whenever flow direction changes, insert the size of packets in the interval), number markers

[3] To use Jaccard's Similarity as a classifier, they turned the metric value into a class membership probability by dividing it by the sum of all metric values in the traininig set.

[4] An interval refers to the time period without packet flow change. Moving from one interval to the other happens when the direction of the flow changes.

(number of packets in every interval), total transmitted bytes, etc. They used Weka tool [20] to fine-tune the SVM parameters. The proposed technique has been evaluated using two experiments: Closed-world and Open-world. In the closed-world experiment, the same set of 775 websites of Federrath et al. [12] as well as ten-fold cross validation have been used to estimate the accuracy. As of open-world experiment, 5000 websites have been randomly chosen among the top one million websites according to Alexa [21] in addition to 5 censored websites. The closed-world experiment showed that the SVM technique resulted in an accuracy of 30% for the basic variant and 54% when all features are used[5]. The open-world experiment showed that, censored websites were successfully identified with a true positives rate between 56 and 73% while the false positives rate was less than 1%.

The most recent contribution was by Cai et al. [15]. As in Panchenko et al. [14], they represented a traffic trace as a sequence of (negative and positive) packet lengths. The training and testing is based on an SVM with a distance-based kernel. They tried several variants of parameters and distances and obtained the best accuracy with a Damerau-Levenshtein edit distance [22, 23]. The use of this distance is motivated by the fact that it is the length of the shortest sequence of character insertions, deletions, substitutions, and transpositions required to transform a trace t to t'. These operations correspond to discarding and reordering of packets in a stream. In order to compute the distance between two traces of different lengths, the Damerau-Levenshtein distance is normalized with respect to the length of the shortest trace between the two. For evaluation they used the top 1000 websites according to Alexa which are then filtered to 800 websites. Using the basic version of Tor, they could successfully identify 80% of visited websites. However, when random cover packets are added to the traffic, the accuracy falls to 50%. Decreasing the size of the training set from 36 to 4 samples decreased the accuracy with 20%.

In all aformentioned works, without exception, the website fingerprinting approaches have been evaluated using only one web browser, mainly, Firefox. Since web browser use different browser engines and user agents and consequentely fetch pages differently, we strongly think that the chosen web browser has an impact on the estimated accuracy. In this paper, we consider a representative set of popular web browsers and repeat the data collection and experiments for every one of them. Our aim is to compare the resistance of web browsers to website fingerprinting attacks.

3 Threat Model

The typical threat model for anonymity systems is a global passive adversary that can observe all the traffic of the network. However, since Tor is a low-latency anonymity system, it has been designed to protect against a weaker form of adversary. Indeed, it is assumed that the adversary can observe only

[5] The feature with the highest contribution was the total number of packets in the traffic trace.

some fraction of the network traffic; who can generate, modify, delete, or delay traffic; who can operate onion routers of his own; and who can compromise some fraction of the onion routers [1]. In this paper we assume a weaker threat model where the attacker can only access the encrypted traffic between the client and the first Tor relay. The attacker does not generate, modify, delete or delay any traffic which makes the attack completely stealth. Except the attacker own Tor node, no other Tor relay or server is compromised which makes the attack easily deployable in practice.

4 Tor Traffic Capture

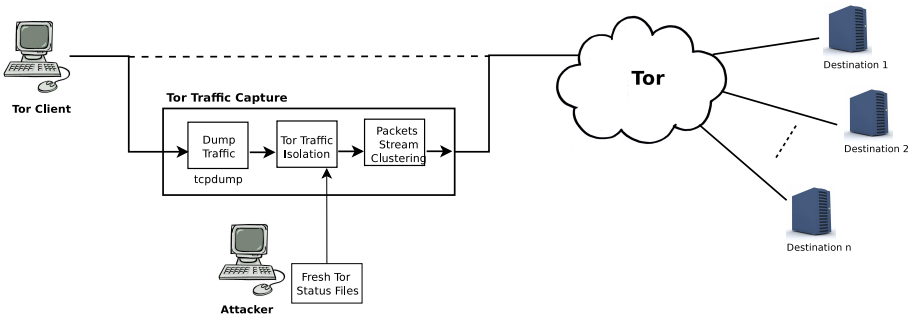


Fig. 1. Tor Traffic Capture

Traffic analysis typically starts by intercepting the traffic packets. In a typical scenario, only the victim and the gateway can capture the data flowing between them. In practice, however, several entities might have access to the traffic packets. The administrator of the LAN has access to the traffic of all endpoints in the network. The ISP (Internet Service Provider) can monitor the traffic of any of its subscribers. A Law Enforcement Agency, after approval from the ISP, can observe and record the traffic of any internet user which is referred to as lawful intercept. A censoring entity can observe the traffic of any user in its "juridiction". In addition to these entities, a malicious user in the LAN can carry out a MITM (Man-In-The-Middle) attack between the victim and the gateway and make all the traffic pass through her. The MITM attack can be easily performed using ARP spoofing/poisoning. Cain & Abel [24] and Ettercap [25] are two popular tools for ARP Spoofing/Poisoning.

Assuming that the attacker can intercept the Tor traffic packets using one of the above scenarios, the Tor traffic capture goes through three stages as shown in Figure 1. First the traffic is dumped in a file using a simple tool like tcpdump[6].

[6] Alternative tools like Wireshark/tshark or Omnipeek can be used as well.

Then the raw traffic is filtered to keep only Tor related traffic. Finally, the traffic is classified into streams.

Since the list of Tor relays is public, we use it to filter Tor traffic from the rest of the traffic. The list of Tor relays can be extracted from the Tor status files, in particular the cached-consensus file. These files can be downloaded manually from one of the authorities or they can be accessed directly in the Tor status local folder. Tor automatically updates those files once they are no more fresh. In our setting, the attacker, which is also a Tor client, uses her own Tor status files to extract the list of Tor relays.

The next step in the Tor traffic capture is to classify the packets into streams at the TCP protocol level[7]. Every stream is then tracked using IP addresses, ports, TCP flag bits, Sequence and Acknowledgment numbers. The stream is closed after a TCP connection termination (FIN, ACK-FIN and ACK).

Typically, Tor creates two types of streams: short living streams and long living streams. Short living streams are streams to download either router descriptors or directory consenus. They last around 3 minutes because the download takes a couple of seconds and then the stream stays idle until it hits the maximum stream idle period which is set to 3 minutes in Tor. Typically, only one circuit is established during a short livinig stream which is a single-hop circuit. The long living streams are for data communications and typically several 3-hop circuits are established during the lifetime of such streams. Besides, since all Tor communications are encrypted, all Tor streams initiate a TLS handshake just after the TCP three-way handshake.

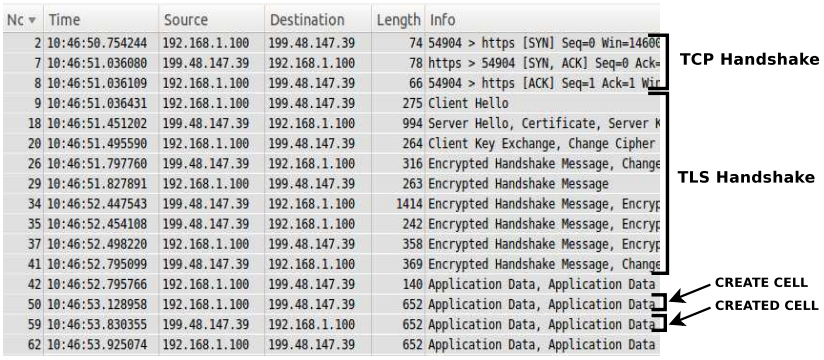


Fig. 2. Tor stream initial packets generated by Wireshark

[7] It is important to note that the classified streams at this stage are streams on top of which Tor communications are conveyed. These are different from the TCP streams tunneled through Tor. Hence, there are two levels of TCP streams: one level below the Tor protocol and one level on top of Tor protocol.

Figure 2 shows the initial packets in a Tor stream. The three first packets are for establishing the TCP connection then a TLS handshake follows. The illustrated stream is a short-living stream whose goal is to download a set of routers descriptors. Therefore, it creates a single hop circuit with the directory server which is performed using a CREATE and CREATED cells as shown in the Figure.

5 Data Collection

In order to evaluate the accuracy of the website fingerprinting for each web browser, 100 websites have been used. The set of 100 websites is composed of 90 randomly chosen websites from the top 1000 visited websites worldwide according to Alexa [21] and 10 censored websites in some countries of the Middle East. All censored websites are related to anonymizers and proxy services[8]. Website traffic traces are collected in 24 hours sessions. In each session, websites are fetched several times in a round-robin fashion. For visiting websites, we used two lab machines running Ubuntu Linux. Another Windows 7 machine is used to fetch websites through Internet Explorer. Traffic packets are dumped using tcpdump version 4.1.1 and libpcap version 1.1.1. Only packet headers are dumped in the dump file[9]. The experiments were performed using Tor version 0.2.2.39.

We wrote a python script to automate the fetching of websites. For each website, the script proceeds as follows: (1) it records the system time, (2) requests the website url, (3) waits for 50 seconds (the time to load the website[10], (4) stops the website connection, (5) records the system time, (6) waits for 10 seconds (to have a time gap between every two visits). Time snapshots are taken just before and after a website visit so that they can be intersected with the dump file in order to isolate the traffic for every website visit.

Once isolated, the taffic corresponding to each visit is represented as a sequence of packet sizes where a positive value refers to an inbound packet while a negative value refers to an outbound packet. This representation captures three features about the trace, namely, the size and direction of each packet and the order of packets[11]. The traffic representation is the same as recent works [14, 15]. For every visit, we keep only the first 500 packets of the traffic so that all obtained sequences have the same length[12].

Interestingly, parsing Tor traffic during a website visit shows that packets are flowing through several TCP streams not just one. One reason is that Tor needs to update the Tor relays status regularly by fetching fresh data from the directory servers and also to send dummy cells to keep some circuits open. TCP streams

[8] Examples of these websites include: torpoject.org, unblock-proxy.net, etc.
[9] tcpdump is launched with options -n -tttt.
[10] If a website takes more than 50 seconds to load, the sample sequence will be incomplete.
[11] As in most of related work, acknowledgement packets are ignored.
[12] This is a requirement for the classification algorithm.

used for fetching directory servers data can be easily distinguished from TCP streams used for data communication since the number of packets exchanged does not exceed 100 packets. Interestingly, ruling out these short TCP streams, the traffic resulting from visiting some websites is carried through two TCP streams with more than 200 packets each. This shows that Tor does not always multiplex the traffic of a website in a single stream. For those website visits, the corresponding traffic sequence is obtained by merging both streams into one. Similarly to normal visits, only the first 500 packets of the merged stream are kept.

In order to avoid the noise introduced by active content (Flash, etc.) and the browser cache, active content and caching are disabled. For instance, Chrome internet browser is used with the "incognito" mode while firefox is used with the "private" mode.

Four Web browsers have been used for fetching websites, namely, Chrome 18.0, Firefox 15.0.1, Internet Explorer 9.0 and Konqueror 4.8.5.

The same experiment is performed four times, each with a different browser. The experiment consists in 10 iterations. Every iteration consists in visiting all 100 websites once. Hence, every website is visited 10 times using the same web browser yielding a maximum of 10 samples for very website[13].

6 Classification Algorithm

The goal of this paper is to compare popular web browsers with respect to their capabilities to resist to web site fingerprinting. To this end, we use a classical classifier, namely, decision trees. It is important to mention that recent related work showed that better fingerprinting results can be obtained using other classifiers in particular Support Vector Machine (SVM) based. The next paragraphs give an overview of the decision tree classifier and the techniques used to evaluate the accuracy of the classifier for every web browser.

6.1 Decision Tree Classifier

Decision tree learning is a well known and classic classification technique [26]. It is very popular because it is self-explanatory, easy to understand and to use since it requires few parameter settings. It has been successfully used for classification in several diverse areas. Overall, it is well suited for exploratory knowledge discovery. In this paper we use a decision tree known as C4.5 [27] to classify website visits traffic sequences.

A decision tree can be learned typically using a top-down approach where each node corresponds to one of the input variables, each edge corresponds to possible values of each variable, and each leaf correspond to a class label. Every data set is split into subsets based on attributed values. This process is repeated

[13] Some website visits resulted in less than 500 packets. These sequences are discarded from the data set.

recursively and is called recursive patitioning. The recursion is completed when splitting adds nothing to the prediction. Inducing a decision tree using a top-down approach requires dealing with three other issues apart from selecting the best attribute to use at each node in the tree. Firstly, one has to choose a splitting threshold to form the two children for each node. Second, one needs a criterion to determine when to stop growing the tree. Thirdly, the final issue is how to decide what class label to assign for the terminal (leaf) node.

Classic strategies for splitting mainly focus on the use of impurity criteria, e.g., information gain, gain ratio and gini index. C4.5 decision tree uses gain ratio to select the best attribute and choose the optimal splitting threshold. In this approach, the attribute value that provides the best gain ratio is chosen as splitting threshold. To address the second issue discussed above, i.e., to avoid difficulties in choosing a stopping rule, most decision tree induction algorithms grow the tree to its maximum size where the terminal nodes are pure or almost pure, and then selectively prune the tree. The class label of each of the terminal nodes are typically decided based on the majority voting, i.e. the class label of data instances that are major in terms of counting compared to the other classes that contain in the respective terminal node.

6.2 Cross-Validation

To achieve a generalized performance of the decision tree used in this paper a cross-validation (CV) scheme is applied. CV is a well known method to test the performance of a classifier by varying training and test datasets [28]. CV is a standard test commonly used to test the ability of the classification system using various combinations of the testing and training data sets [29, 28, 30]. In this method, classification is measured by systematically excluding some data instances during the training process and testing the trained model using the excluded instances [31]. The process is repeated to cover all the dataset as testing dataset. In this paper, we have chosen 10-fold CV scheme where each time data in 1 fold are applied as test data and the rest 9 folds are used to train the model.

6.3 Performance Metric

Classification accuracy is one of the widely used performance metric to evaluate a classifier. Classification accuracy (ACC) is defined as the ratio of the number of all samples that are classified correctly over the total number of samples available (N).

$$ACC = (TP + TN)/N \tag{1}$$

where, TP (True Positives) = the total data instances from positive class that are classified as positive by the classifier; TN (True Negatives) = the total data instances from negative class that are classified as negative by the classifier.

7 Web Browser Resistance to Fingerprinting

Popular web browsers differ in several aspects, in particular, they use different web browser engines. The engine does most of the work of a web browser since it retrieves the document corresponding to a given URL and handles links, cookies, scripting, plug-ins loading, etc. The type of the web browser engine has an impact on the shape of the observed (encrypted) traffic. For example, some web browser engines may wait until all data is received before rendering a page while others may begin rendering before all data is received.

In order to compare the resistance of popular web browsers to website fingerprinting attacks, data is collected using different web browsers and then a C4.5 decision tree classifier is used to evaluate the accuracy of the website fingerprinting. More precisely, once the data about website visits is collected, we evaluated the accuracy of website fingerprinting in four scenarios:

- **Basic Packets Sequence**: The traffic trace is the first 500 packets of the TCP stream with the largest number of packets.
- **Merged Streams**: The traffic trace is the first 500 packets obtained by merging all TCP streams with more than 200 packets. If only one TCP stream has more than 200 packets, this scenario is the same as the first one.
- **Rounded Packet Sizes**: The same as the first scenario but the packet size values are rounded to multiples of 600. For instance, a packet size of 512 is rounded to 600 while a packet size of 743 is rounded to 1200.
- **Merged Streams and Rounded Packet Sizes**: This scenario is the combination of the two previous scenarios.

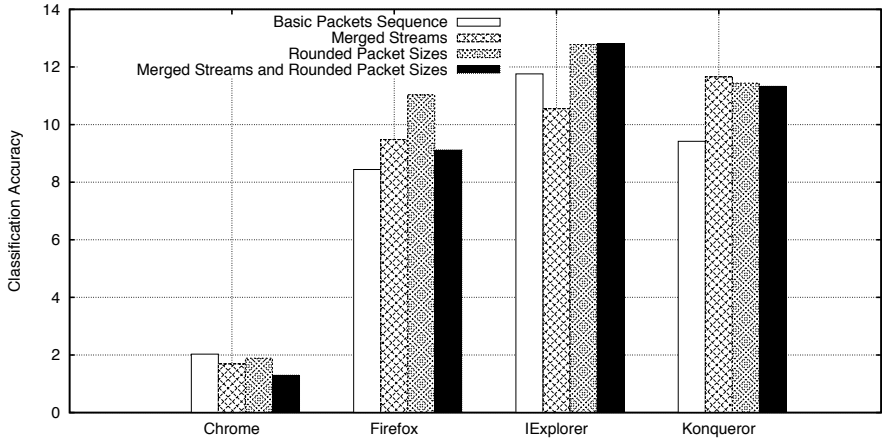


Fig. 3. Website Fingerprinting Accuracy for Common Web Browsers

Figure 3 shows the accuracy of the C4.5 classifier for each browser and for each scenario. Using Firefox, Internet Explorer and Konqueror, more than 9% of websites have been correctly identified by the classifier. With Chrome, however, only 2% of websites have been successfully identified. The histogram shows also that Rounding packet size values improves the efficiency of our classifier for all browsers. Merging TCP streams, on the other hand, improved the efficiency of the classifier only for Firefox and Konqueror. Merging streams resulted in lower accuracy than the basic scenario for Chrome and Internet Explorer. The most important result illustrated by the histogram is that Chrome browser offers a better resistance to website fingerprinting than the other studied browsers. The advantage Chrome browser has on the other studied browsers is expected to be preserved in case a more efficient classifier (e.g. [15]) is used.

8 Conclusion

Website fingerprinting is a new attack on Tor anonymity system that tries to reveal the identities of visited websites by recognizing patterns in the Tor traffic. Compared to previous attacks on Tor, in particular confirmation attacks, website fingerprinting does not require an attacker with extended capabilities. Only the ability to sniff the Tor client encryted traffic is required. In this paper we presented a detailed survey on website fingerprinting techniques which recently reached high accuracy rates (80%) [15] on Tor anonymity system. The main contribution of this paper, however, is an empirical analysis of how much resistance popular web browsers provide against website fingerprinting. Four notable web browsers have been considered, namely, Firefox, Chrome, Internet Explorer, and Konqueror. The analysis showed that the resistance of Chrome to website fingerprinting is five times better than the remaining web browsers. Since most of existing fingerprinting techniques have been evaluated using Firefox web browser [12, 14, 15], we expect the accuracy results to be reduced in case Chrome browser is used.

There are two main mitigation approaches for the website fingerprinting attack. The first and the most commonly used approach is padding where the sender appends some random (dummy) bits to the actual data to obtain, for instance, fixed size packets. It has been shown that padding reduces the accuracy of fingerprinting techniques only slightly [14, 15]. The second mitigation approach is traffic camouflage which can be implemented in two ways. The first variant is to obfuscate the actual traffic by loading several pages simultaneously. Panchenko et al. [14] load a randomly chosen page whenever an actual website is to be visited. The second variant is to disguise the actual traffic within a typical encrypted cover protocol such as Skype voice over IP. The StegoTorus proxy [32] constructs a database of pre-recorded packet traces from real sessions of a given cover protocol. Then, when a Tor client visits a website, it choses randomly a pre-recorded trace from the database which is used as a template to reshape the actual traffic. The packet sizes of the pre-recorded trace are matched exactly and the packet timings are matched "to the nearest millisecond".

References

1. Dingledine, R., Mathewson, N., Syverson, P.: Tor : the second-generation onion router. In: Proceedings of the 13th Usenix Security Symposium. (August 2004)
2. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: A system for anonymous and unobservable Internet access. In: Proceedings of Designing Privacy Enhancing Technologies, Springer-Verlag, LNCS 2009 (July 2000) 115–129
3. Murdoch, S., Danezis, G.: Low-cost traffic analysis of Tor. In: Proceedings of the 2005 IEEE Symposium on Security and Privacy, IEEE CS (May 2005)
4. Murdoch, S.: Hot or not: Revealing hidden services by their clock skew. In: Proceedings of CCS 2006. (October 2006)
5. Hopper, N., Vasserman, E., Chan-Tin, E.: How much anonymity does network latency leak? ACM Transactions on Information and System Security **13**(2) (February 2010)
6. Evans, N., Dingledine, R., Grothoff, C.: A practical congestion attack on tor using long paths. In: Proceedings of the 18th USENIX Security Symposium. (August 2009)
7. Mittal, P., Khurshid, A., Juen, J., Caesar, M., Borisov, N.: Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting. In: Proceedings of the 18th ACM conference on Computer and communications security. CCS '11, New York, NY, USA, ACM (2011) 215–226
8. Hintz, A.: Fingerprinting websites using traffic analysis. In: Privacy Enhancing Technologies (PETS),LNCS. Volume 2482., Springer (2002) 171–178
9. Sun, Q., Simon, D.R., Wang, Y.M., Russell, W., Padmanabhan, V.N., Qiu, L.: Statistical identification of encrypted web browsing traffic. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy. SP '02, Washington, DC, USA, IEEE Computer Society (2002) 19–
10. Bissias, G.D., Liberatore, M., Jensen, D., Levine, B.N.: Privacy vulnerabilities in encrypted http streams. In: Proceedings of the 5th international conference on Privacy Enhancing Technologies. PET'05, Berlin, Heidelberg, Springer-Verlag (2006) 1–11
11. Liberatore, M., Levine, B.N.: Inferring the source of encrypted http connections. In: Proceedings of the 13th ACM conference on Computer and communications security. CCS '06, New York, NY, USA, ACM (2006) 255–263
12. Herrmann, D., Wendolsky, R., Federrath, H.: Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In: Proceedings of the 2009 ACM workshop on Cloud computing security. CCSW '09, New York, NY, USA, ACM (2009) 31–42
13. Shi, Y., Matsuura, K.: Fingerprinting attack on the tor anonymity system. In Qing, S., Mitchell, C., Wang, G., eds.: Information and Communications Security. Volume 5927 of Lecture Notes in Computer Science., Springer Berlin Heidelberg (2009) 425–438
14. Panchenko, A., Niessen, L., Zinnen, A., Engel, T.: Website fingerprinting in onion routing based anonymization networks. In: Proceedings of the 10th annual ACM workshop on Privacy in the electronic society. WPES '11, New York, NY, USA, ACM (2011) 103–114
15. Cai, X., Zhang, X.C., Joshi, B., Johnson, R.: Touching from a distance: website fingerprinting attacks and defenses. In: Proceedings of the 2012 ACM conference on Computer and communications security. CCS '12, New York, NY, USA, ACM (2012) 605–616

16. Safe Web: Safeweb proxy. "http://www.safeweb.com"
17. Rijsbergen, C.J.V.: Information Retrieval. 2nd edn. Butterworth-Heinemann, Newton, MA, USA (1979)
18. Bracewell, R.: Pentagram Notation for Cross Correlation. The Fourier Transform and Its Application. McGraw-Hill, New York, USA (1965)
19. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Second edn. Morgan Kaufmann (June 2005)
20. Weka Tool: Weka: Data mining software in java. "www.cs.waikato.ac.nz/ml/weka"
21. Alexa Website: Alexa: The web information company. "www.alexa.com"
22. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklay (1966)
23. Navarro, G.: A guided tour to approximate string matching. ACM Computing Surveys **33** (1999) 2001
24. Massimiliano Montoro: Cain & abel. "http://www.oxid.it/cain.htm"
25. ALor and NaGA: Ettercap. "http://ettercap.sourceforge.net"
26. Kotsiantis, S.B.: Supervised machine learning: A review of classification techniques. Informatica **31** (2007) 249–268
27. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
28. Barton, J., Lees, A.: An application of neural networks for distinguishing gait patterns on the basis of hip-knee joint angle diagrams. Gait & Posture **5**(1) (1997) 28 – 33
29. Ding, C.H.Q., Dubchak, I.: Multi-class protein fold recognition using support vector machines and neural networks. Bioinformatics **17** (2001) 349–358
30. Hassan, M.R., Begg, R., Taylor, S., Kumar, D.K.: Hmm-fuzzy model for recognition of gait changes due to trip-related falls. In: Proceeding of the IEEE Eng Med Biol Soc., Springer Berlin Heidelberg (2006) 1216–1219
31. Begg, R.K., Palaniswami, M., Owen, B.: Support vector machines for automated gait classification. IEEE Transactions on Biomedical Engineering **52** (2005) 828–838
32. Weinberg, Z., Wang, J., Yegneswaran, V., Briesemeister, L., Cheung, S., Wang, F., Boneh, D.: Stegotorus: a camouflage proxy for the tor anonymity system. In: Proceedings of the 2012 ACM conference on Computer and communications security, New York, NY, USA, ACM (2012) 109–120

Secrecy by Witness-Functions

Jaouhar Fattahi[1], Mohamed Mejri[1] and Hanane Houmani[2]

[1]LSI Group, Laval University, Quebec, Canada
[2] University Hassan II, Morocco

Abstract. In this paper, we introduce a new type of functions to analyze cryptographic protocols statically for the property of secrecy: the Witness-Functions. A Witness-Function is a reliable protocol-dependent function intended to prove the correctness of a protocol through its growth. It bases its calculation on the static part of a message in a role-based specification and ignores the dynamic one by introducing the notion of derivative messages. It offers two interesting bounds that enable an analysis of protocols on an unbounded number of sessions. We give here the way to build these functions and we state the theorem of protocol analysis with the Witness-Functions.

Keywords: Protocol, role-based specification, secrecy, sufficient condition.

1 Motivation and background

In this paper, we introduce a new static approach to analyze cryptographic protocols with witness-functions for the property of secrecy. Intuitively, an increasing protocol preserves the secret. In other words, if the security of any atomic message does not decrease between receiving and sending steps of a protocol, the secret is never leaked. For that, we should define "good" metrics to evaluate the security of any atomic message. This way of thinking has been considered in some previous works. In [1], Steve Schneider proposed the concept of rank-functions as metrics to analyze protocols in CSP algebra. These functions were successful in analyzing Needham-Schroeder protocol. However, a such analysis requires the protocol implementation in CSP [2,3]. Besides, building rank-functions is not a trivial job and their existence is not sure [4]. In [5] Abadi, using Spi-Calculus [6,7], guarantees that: "If a protocol typechecks, then it does not leak its secret inputs". To do so, he requests from the exchanged messages to be composed of four parts having strictly the following types: {secret, public, any, confounder} in order to recognize the security level of every part. However, this approach cannot analyze protocols that had been implemented with no respect to this restriction. In the same vein, Houmani and al. [8,9,10,11] defined universal functions called interpretation functions able to analyze a protocol statically and operate on an abstraction of the protocol called generalized roles, that generate a space of messages with variables. An interpretation function must meet some sufficient conditions to be reliable for the analysis. Obviously, less we have conditions on functions, more we have functions and more we have chance to get protocols proven correct since one function may fail to prove the growth of a protocol but another may manage to do. However, we notice that the conditions on functions were so restrictive that only two concrete functions had been proposed. We believe that the condition related to the full-invariance by substitution, which is the property-bridge between an analysis run on messages of the generalized roles and the conclusion made on valid traces, is the most restrictive one. Since the aim of our approach is to build as more reliable functions as we are able to do, we think that if we free a function from this restrictive condition, we can build more functions. Nevertheless, freeing a function from a condition may impel us to take additional precautions. In this work, we introduce the witness-functions to analyze cryptographic protocols. We show how to build them. We show that a witness-function offers two bounds that allow us to get rid of the restrictive condition of full-invariance by substitution in Houmani's work by using derivation techniques. We state finally the theorem of protocol analysis with the Witness-Functions that sets an interesting criterion for the protocol correctness.

Notations

Hereafter, we give some definitions and conventions that will be used throughout the paper.

+ We denote by $\mathcal{C} = \langle \mathcal{M}, \xi, \models, \mathcal{K}, \mathcal{L}^{\sqsupseteq}, \ulcorner . \urcorner \rangle$ the context containing the parameters that affect the analysis of a protocol:
  - $\mathcal{M}$: is a set of messages built from the algebraic signature $\langle \mathcal{N}, \Sigma \rangle$ where $\mathcal{N}$ is a set of atomic names (nonces, keys, principals, etc.) and Σ is a set of allowed functions (enc: encryption, dec: decryption, $pair$: concatenation (denoted by "." here), etc.). i.e. $\mathcal{M} = T_{\langle \mathcal{N}, \Sigma \rangle}(\mathcal{X})$. We use Γ to denote the set of all possible substitution from $\mathcal{X} \rightarrow \mathcal{M}$. We denote by $\mathcal{A}$ all atomic messages in $\mathcal{M}$, by $\mathcal{A}(m)$ the set of atomic messages (or atoms) in m and by $\mathcal{I}$ the set of agents (principals) including the intruder I. We denote by k^{-1} the reverse key of a key k and we consider that $(k^{-1})^{-1} = k$.
  - ξ : is the equational theory that describes the algebraic properties of the functions in Σ by equations. e.g. $dec(enc(x, y), y^{-1}) = x$.
  - $\models$: is the inference system of the intruder under the equational theory. Let M be a set of messages and m a message. $M \models m$ means that the intruder is able to infer m from M using her capacity. We extend this notation to traces as following: $\rho \models m$ means that the intruder can infer m from the messages exchanged in the trace ρ.
  - $\mathcal{K}$: is a function from $\mathcal{I}$ to $\mathcal{M}$, that assigns to any agent (principal) a set of atomic messages describing her initial knowledge. We denote by $K_{\mathcal{C}}(I)$ the initial knowledge of the intruder, or simply $K(I)$ where the context is clear.
  - $\mathcal{L}^{\sqsupseteq}$: is the security lattice $(\mathcal{L}, \sqsupseteq, \sqcup, \sqcap, \bot, \top)$ used to attribute security levels to messages. A concrete example of a lattice is $(2^{\mathcal{I}}, \subseteq, \cap, \cup, \mathcal{I}, \emptyset)$ that will be used to attribute to a message α the set of principals that are allowed to know it.
  - $\ulcorner . \urcorner$: is a partial function that assigns a value of security (type) to a message in $\mathcal{M}$. Let M be a set of messages and m a message. We write $\ulcorner M \urcorner \sqsupseteq \ulcorner m \urcorner$ if $\exists m' \in M . \ulcorner m' \urcorner \sqsupseteq \ulcorner m \urcorner$

+ Let p be a protocol, we denote by $R_G(p)$ the set of the generalized roles extracted from p. A generalized role is a protocol abstraction where the emphasis is put on a particular principal and all the unknown messages, and on which no verification could be done by the agent, are replaced by variables. A generalized role ends always by a sending step and an intruder I is introduced to describe that the received messages and the sent messages are probably sent or received by the intruder. More details about the role-based specification are in [12,13,14]. Hereafter, an example of a variation of NSL protocol written in a role-based specification:

$$
\begin{array}{l}
m_1 : A \longrightarrow B : \{N_a.A\}_{k_b} \\
m_2 : B \longrightarrow A : \{B.N_a\}_{k_a}.\{B.N_b\}_{k_a} \\
m_3 : A \longrightarrow B : A.B.\{N_b\}_{k_b}
\end{array}
$$

Table 1. A variation of NSL protocol

The generalized roles of this protocol in a role-based specification are $\mathcal{R}_{\mathcal{G}}(p_{NSL})$= $\{A^{\mathcal{1}}_{\mathcal{G}}, A^{\mathcal{2}}_{\mathcal{G}}, B^{\mathcal{1}}_{\mathcal{G}}, B^{\mathcal{2}}_{\mathcal{G}}\}$ where:

$$
\begin{array}{lll}
A^{\mathcal{1}}_{\mathcal{G}} = & i.1\ A & \longrightarrow I(B) : \{N^i_a.A\}_{k_b} \\
A^{\mathcal{2}}_{\mathcal{G}} = & i.1\ A & \longrightarrow I(B) : \{N^i_a.A\}_{k_b} \\
 & i.2\ I(B) & \longrightarrow A \quad : \{B.N^i_a\}_{k_a}.\{B.X\}_{k_a} \\
 & i.3\ A & \longrightarrow I(B) : A.B.\{X\}_{k_b} \\
 & & \\
B^{\mathcal{1}}_{\mathcal{G}} = & i.1\ I(A) & \longrightarrow B \quad : \{Y.A\}_{k_b} \\
 & i.2\ B & \longrightarrow I(A) : \{B.Y\}_{k_a}.\{B.N^i_b\}_{k_a} \\
B^{\mathcal{2}}_{\mathcal{G}} = & i.1\ I(A) & \longrightarrow B \quad : \{Y.A\}_{k_b} \\
 & i.2\ B & \longrightarrow I(A) : \{B.Y\}_{k_a}.\{B.N^i_b\}_{k_a} \\
 & i.3\ I(A) & \longrightarrow B \quad : A.B.\{N^i_b\}_{k_b}
\end{array}
$$

We denote by $\mathcal{M}_p^{\mathcal{G}}$ the set of messages with variables generated by $R_G(p)$, by $\mathcal{M}_p$ the set of closed messages generated by substituting terms in $\mathcal{M}_p^{\mathcal{G}}$. We denote by $R+$ (respectively R^-) the set of sent messages (respectively received messages) by a honest agent in the role R. Commonly , we reserve the uppercase letters for sets or sequences of elements and the lowercase for single elements. For instance M denotes a set of messages, m a single message, R a role composed of a sequence of steps, r a step and $R.r$ the role ending by the step r.

+ A valid trace is an interleaving of instantiated generalized roles where each message sent by the intruder can be produced by her using her capacity and the previous received messages. We denote by $[\![p]\!]$ the set of valid traces of p.
+ We assume that the intruder has the full-control of the net, as described in the Dolev-Yao model [15] with no restriction neither on the size of messages nor on the number of sessions.

2 Increasing protocols are correct with respect to the secrecy property

To analyze a protocol, we need reliable functions to estimate the security level of every atomic message. In this section, we state sufficient conditions allowing to guarantee that a function is reliable. We prove that an increasing protocol is correct with respect to the secrecy property when analyzed with such functions.

2.1 $\mathcal{C}$-reliable interpretation functions

Definition 21 *(Well-formed interpretation function)*
Let F be an interpretation function.
F is well-formed in $\mathcal{C}$ if:
$\forall M, M_1, M_2 \subseteq \mathcal{M}, \forall \alpha \in \mathcal{A}(\mathcal{M})$*:*

$$\begin{cases} F(\alpha, \{\alpha\}) & = \bot \\ F(\alpha, M_1 \cup M_2) & = F(\alpha, M_1) \sqcap F(\alpha, M_2) \\ F(\alpha, M) & = \top, \ \textit{if } \alpha \notin \mathcal{A}(M) \end{cases}$$

For an atom α in a set of messages M, a well-formed interpretation function returns the bottom value "$\bot$" if α appears in clear in M. It returns for it in the union of two sets, the minimum "$\sqcap$" of the two values calculated in each set seperately. It returns the top value "$\top$" if α does not appear at all in M.

Example 22 *Let $m_1 = \{\alpha\}, m_2 = \{\beta.A\}_{k_{ab}}$, $M = \{m_1, m_2\}$ and F a well-formed interpretation function.*
• $F(\alpha, M) = F(\alpha, \{m_1\} \cup \{m_2\}) = F(\alpha, \{\alpha\}) \sqcap F(\alpha, \{\beta.A\}_{k_{ab}}) = \bot \sqcap \top = \bot$
• $F(\beta, M) = F(\alpha, \{m_1\} \cup \{m_2\}) = F(\beta, \{\alpha\}) \sqcap F(\beta, \{\beta.A\}_{k_{ab}}) = \top \sqcap F(\beta, \{\beta.A\}_{k_{ab}})$
$= F(\beta, \{\beta.A\}_{k_{ab}})$

Definition 23 *(Full-invariant-by-intruder interpretation function)*
Let F be an interpretation function.
F is full-invariant-by-intruder in $\mathcal{C}$ if:
$\forall M \subseteq \mathcal{M}, m \in \mathcal{M}. M \models_{\mathcal{C}} m \Rightarrow \forall \alpha \in \mathcal{A}(m). (F(\alpha, m) \sqsupseteq F(\alpha, M)) \vee (\ulcorner K(I) \urcorner \sqsupseteq \ulcorner \alpha \urcorner)$

An interpretation function F is said to be full-invariant-by-intruder if when it assigns a security level to an atomic message α in a set of messages M, this level cannot be decreased by the intruder. That is to say, the intruder cannot infer another message m from M in which the level of security of α decreases (i.e. $F(\alpha, m) \sqsupseteq F(\alpha, M)$) using her capacity in the context of verification, unless α is initially intended to be known by the intruder (i.e. $\ulcorner K(I) \urcorner \sqsupseteq \ulcorner \alpha \urcorner$).

Definition 24 *(Reliable interpretation function)*
Let F be an interpretation function and $\mathcal{C}$ be a context.

$$F \text{ is } \mathcal{C}\text{-reliable if } F \text{ is well-formed and } F \text{ is full-invariant-by-intruder in } \mathcal{C}.$$

Definition 25 *[F-increasing protocol]*
Let F be an interpretation function and p a protocol.
p is F-increasing in $\mathcal{C}$ if:
$\forall R.r \in R_G(p), \forall \sigma \in \Gamma : \mathcal{X} \to \mathcal{M}_p$ *we have:*

$$\forall \alpha \in \mathcal{A}(\mathcal{M}_p).F(\alpha, r^+\sigma) \sqsupseteq \ulcorner \alpha \urcorner \sqcap F(\alpha, R^-\sigma)$$

A F-increasing protocol is a protocol where every involved principal (every substituted generalized role) never decreases the security levels of received components. When a protocol is F-increasing and F is a reliable function, it is intuitively easy to prove its correctness with respect to the secrecy property. In fact, if every agent appropriately protects her sent messages (if she initially knows the security level of a component, she has to encrypt it with at least one key having a similar or higher security level, and if she does not know its security level, she estimates it using a reliable function F), the intruder can never reveal it.

Definition 26 *(Secret disclosure)*
Let p be a protocol and C a context.
We say that p discloses a secret $\alpha \in \mathcal{A}(\mathcal{M})$ in $\mathcal{C}$ if:

$$\exists \rho \in [\![p]\!].(\rho \models_{\mathcal{C}} \alpha) \wedge (\ulcorner K(I) \urcorner \not\sqsupseteq \ulcorner \alpha \urcorner)$$

A secret disclosure consists in exploiting a valid trace of the protocol (denoted by $[\![p]\!]$) by the intruder using her knowledge $K(I)$ in the context of verification $\mathcal{C}$, to infer a secret α that she is not allowed to know (expressed by: $\ulcorner K(I) \urcorner \not\sqsupseteq \ulcorner \alpha \urcorner$ in the definition 26).

Lemma 27
Let F be a $\mathcal{C}$-reliable interpretation function and p a F-increasing protocol.
We have:

$$\forall m \in \mathcal{M}.[\![p]\!] \models_{\mathcal{C}} m \Rightarrow \forall \alpha \in \mathcal{A}(m).(F(\alpha, m) \sqsupseteq \ulcorner \alpha \urcorner) \vee (\ulcorner K(I) \urcorner \sqsupseteq \ulcorner \alpha \urcorner).$$

Proof. See the proof 819 in [16] ■

The lemma 27 expressess an intuitive result that for any atom α in a message generated by an increasing protocol, its security level calculated by a reliable interpretation function is maintained greater than its initial value in the context, if the intruder is not initially allowed to know it. Thus, initially the atom has a certain security level. This value cannot be decreased by the intruder using her initial knowledge and received messages since a reliable function is full-invariant by intruder. In each new step of any valid trace, involved messages are better protected since the protocol is increasing. The proof is then run by induction on the size of the trace and uses the reliability properties of the interpretation function in every step.

Theorem 21 *(Correctness of increasing protocols)*
Let F be a $\mathcal{C}$-reliable interpretation function and p a F-increasing protocol.

$$p \text{ is } \mathcal{C}\text{-correct with respect to the secrecy property.}$$

Proof. See the proof 820 in [16] ■

In this section, we have shown that an increasing protocol is correct with respect to the secrecy property when analyzed with an interpretation function that is full-invariant by intruder and well-formed, or simply reliable. Please notice that compared to the sufficient conditions stated in [11], we have one less. Houmani in [11] requested that a protocol must be increasing on the messages of the generalized roles of the protocol (that contain variables), and demanded from the interpretation function to resist to the problem of substitution of variables. Here, we free our functions from this restrictive condition in order to be able to build more functions. We relocate this condition in our new definition of an increasing protocol, that is requested now to be increasing on valid traces. The problem of substitution migrates to the protocol and becomes less harder to deal with.

3 Building reliable interpretation functions

As seen in the previous section, to analyze a protocol statically we need reliable interpretation functions to evaluate the level of security of each atom in a message. In this section, we propose a constructive way to build these functions. We first show how to build a generic class of reliable selections inside the protection of the most external key (or simply the external key), then we provide specialized selections that are instances of this class, and finally we show the way to build reliable selection-based interpretation functions. Similar techniques have been used in some previous works and especially in [8,10,11] to build functions based on the direct key of encryption and in [17] to verify correspondences for security in protocols. But first, we introduce the notion of well-protected messages that have interesting properties that we will use in the definition of reliable selections.

3.1 Well-protected messages

We denote by $\mathcal{E}_\mathcal{C}$ the set of encryption functions and by $\overline{\mathcal{E}}_\mathcal{C}$ the complementary set $\Sigma \backslash \mathcal{E}_\mathcal{C}$ in a context of verification $\mathcal{C}$. In the definition 31, we define the application $keys$ that returns the encryption keys of any atom α in a message m.

Definition 31 *(Keys)*
Let $M \subseteq \mathcal{M}$, $f \in \Sigma$ and $m \in M$.
We define the application Keys as follows:

$$Keys : \mathcal{A} \times \mathcal{M} \longrightarrow \mathcal{P}(\mathcal{P}(\mathcal{A}))$$

$\forall t_1, t_2 ... t_n$ subterms of m:

$$\begin{array}{ll} Keys(\alpha, \alpha) & = \{\emptyset\} \\ Keys(\alpha, \beta) & = \emptyset, \textit{ if } \alpha \neq \beta \textit{ and } \beta \in \mathcal{A} \\ Keys(\alpha, f_k(t_1, ..., t_n)) & = \{k\} \otimes \bigcup_{i=1}^{n} Keys(\alpha, t_i), \textit{ if } f_k \in \mathcal{E}_\mathcal{C} \\ Keys(\alpha, f(t_1, ..., t_n)) & = \bigcup_{i=1}^{n} Keys(\alpha, t_i), \textit{ if } f \in \overline{\mathcal{E}}_\mathcal{C} \end{array}$$

We extend the application $Keys$ to sets as follows:

$$\forall M \subseteq \mathcal{M}. Keys(\alpha, M) = \bigcup_{m \in M} Keys(\alpha, m) \textit{ and } Keys(\alpha, \emptyset) = \emptyset.$$

Definition 32 *(Equational theory, rewriting system and normal form)*
We assume that we can transform the equational theory ξ given in the context of verification to a convergent rewriting system $\rightarrow_\xi$ such that:

$$\forall m \in \mathcal{M}, \forall \alpha \in \mathcal{A}(m), \forall l \rightarrow r \in \rightarrow_\xi, \qquad Keys(\alpha, r) \subseteq Keys(\alpha, l) \tag{3.1.1}$$

We denote by $m_\Downarrow$ the normal form of m in $\rightarrow_\xi$.

The normal form of a message in the definition 32 is the one that has the smallest set of encryption keys and eliminates all the unnecessary keys (e.g. $e(k, d(k^{-1}, m)) \rightarrow m$). This kind of rewriting systems orientation poses no problem with the most of equational theories [18,19,20].

In the definition 33 we introduce the application $Access$ where every element of $Access(\alpha, m)$ contains a set of required keys to decrypt α in m after elimination of unnecessary keys by the normal form defined in 32.

Definition 33 *(Access)*
Let $M \subseteq \mathcal{M}$, $f \in \Sigma$ and $m \in M$.
We define the application Accessor as follows:

$$Access : \mathcal{A} \times \mathcal{M} \longrightarrow \mathcal{P}(\mathcal{P}(\mathcal{A}))$$

$\forall t_1, t_2 ... t_n$ subterms of m:

$$\begin{array}{ll}
Access(\alpha, \alpha) & = \{\emptyset\} \\
Access(\alpha, \beta) & = \emptyset, \text{ if } \alpha \neq \beta \text{ and } \beta \in \mathcal{A} \\
Access(\alpha, f_k(t_1, ..., t_n)) & = \{k^{-1}\} \otimes \overset{n}{\underset{i=1}{\cup}} Access(\alpha, t_i), \text{ if } f_k \in \mathcal{E}_\mathcal{C} \text{ and } f_k(t_1, ..., t_n) = f_k(t_1, ..., t_n)_{\Downarrow} \\
Access(\alpha, f(t_1, ..., t_n)) & = \overset{n}{\underset{i=1}{\cup}} Access(\alpha, t_i), \text{ if } f \in \overline{\mathcal{E}}_\mathcal{C} \text{ and } f_k(t_1, ..., t_n) = f_k(t_1, ..., t_n)_{\Downarrow} \\
Access(\alpha, f(t_1, ..., t_n)) & = Access(\alpha, f(t_1, ..., t_n)_{\Downarrow}), \text{ if not.}
\end{array}$$

We extend the application $Access$ to sets as follows:

$$\forall M \subseteq \mathcal{M}. Access(\alpha, M) = \underset{m \in M}{\cup} Access(\alpha, m) \text{ and } Access(\alpha, \emptyset) = \emptyset.$$

Example 34 *Let m be a message such that:* $m = \{\{A.D.\alpha\}_{k_{ab}}.\alpha.\{A.E.\{C.\alpha\}_{k_{ef}}\}_{k_{ab}}\}_{k_{ac}}$;
$Access(\alpha, m) = \{\{k_{ac}^{-1}, k_{ab}^{-1}\}, \{k_{ac}^{-1}\}, \{k_{ac}^{-1}, k_{ab}^{-1}, k_{ef}^{-1}\}\}$.

In the definition 35, we define a well-protected message. Informally, a well-protected message is a message such that every atom α in it such that $\ulcorner \alpha \urcorner \sqsupset \perp$ is encrypted by at least one key k such that $\ulcorner k^{-1} \urcorner \sqsupseteq \ulcorner \alpha \urcorner$ after elimination of unnecessary keys by the normal form defined in 32.

Definition 35 *(Well-protected message)*
Let $\mathcal{C}$ be context of verification, $m \in \mathcal{M}$, $M \subseteq \mathcal{M}$ and $\alpha \in \mathcal{A}(m)$ such that $\ulcorner \alpha \urcorner \sqsupset \perp$.

We say that α is well-protected in m if:

$$\forall K \in Access(\alpha, m). \ulcorner K \urcorner \sqsupseteq \ulcorner \alpha \urcorner$$

We say that α is well-protected in M if:

$$\forall m \in M. \alpha \text{ is well-protected in } m$$

We say that m is well-protected in $\mathcal{C}$ if:

$$\forall \alpha \in \mathcal{A}(m). \alpha \text{ is well-protected in } m$$

We say that M is well-protected in $\mathcal{C}$ if:

$$\forall m \in M. m \text{ is well-protected in } \mathcal{C}.$$

In the definition 36, we define $Clear(m)$. Informally, $Clear(m)$ is the set of all atoms that appear in clear in m after elimination of unnecessary keys by the normal form defined in 32.

Definition 36 *(Clear)*
Let $m \in \mathcal{M}$ and $M \subset \mathcal{M}$.
$Clear(m) = \{\alpha \in \mathcal{A}(m) |\ \emptyset \in Access(\alpha, m)\}$
We extend this definition to sets as follows:

$$Clear(M) = \underset{m \in M}{\cup} Clear(m)$$

Lemma 37
Let M be a set of well-protected messages in $\mathcal{M}$. We have:

$$M \models_{\mathcal{C}} m \Rightarrow \forall \alpha \in \mathcal{A}(m).(\alpha \textit{ is well-protected in } m) \vee (\ulcorner K(I) \urcorner \sqsupseteq \ulcorner \alpha \urcorner)$$

Proof. See the proof 912 in [16] ■

The lemma 37 states that from a set of well-protected messages, all atomic messages beyond the knowledge of the intruder (i.e. $\ulcorner K(I) \urcorner \not\sqsupseteq \ulcorner \alpha \urcorner$) remain well-protected in any message that the intruder could infer. In fact, since we operate on well-protected messages, every atom such that $\ulcorner \alpha \urcorner \sqsupset \bot$ is encrypted by at least one key k such that $\ulcorner k^{-1} \urcorner \sqsupseteq \ulcorner \alpha \urcorner$. So the intruder must retrieve k^{-1} before she sees α not well-protected in any message. Yet, the key k^{-1}, if it appears in M, should be in its turn encrypted by at least one key k' such that $\ulcorner k'^{-1} \urcorner \sqsupseteq \ulcorner k^{-1} \urcorner$ since we operate on well-protected messages. The proof is then run by induction on the encryption keys.

Lemma 38 *(Lemma of non-disclosure of atomic secrets in well-protected messages)*
Let M be a set of well-protected messages in $\mathcal{M}$ and α an atomic message in M.
We have:

$$M \models_{\mathcal{C}} \alpha \Rightarrow \ulcorner K(I) \urcorner \sqsupseteq \ulcorner \alpha \urcorner.$$

Proof. See the proof 914 in [16] ■

3.2 Well-protected messages and increasing protocols

The lemma 38 expresses an important result. It states that a set of well-protected messages never discloses a secret. Thus, an intruder cannot infer from this set of messages something that she is not initially allowed to know. The notion of well-protected messages is linked to the notion of increasing protocols. Indeed, having a reliable interpretation function F and a protocol p, p could not be F-increasing if it does not operate on a set of well-protected messages $\mathcal{M}_p$. Thus, a F-increasing protocol does not leak secrets as established by the theroem 21. This could not happen if $\mathcal{M}_p$ is not well-protected. Thus, an atomic message that is not well-protected in $\mathcal{M}_p$ could be deduced by agents that have the decryption keys even if they are not intented to know it. To be coherent, a reliable interpretation function must give for a non-well-protected atom α in any message m in $\mathcal{M}_p$ a value less than $\ulcorner \alpha \urcorner$ (e.g. $F(\alpha, m) = \bot$).

3.3 Reliable selections in well-protected messages

Now, we will focus on building selections such that when they are composed to suitable homomorphisms, provide reliable interpretation functions. The definition 39 introduces the notion of a well-formed selection and the definition 310 introduces the notion of a full-invariant-by-intruder selection.

Definition 39 *(Well-formed selection)*
Let $M, M_1, M_2 \subseteq \mathcal{M}$ such that M, M_1 and M_2 are well-protected.
Let $S : \mathcal{A} \times \mathcal{M} \longmapsto 2^{\mathcal{A}}$ be a selection.
We say that S is well-formed in $\mathcal{C}$ if:

$$\begin{cases} S(\alpha, \{\alpha\}) & = \mathcal{A}, \\ S(\alpha, M_1 \cup M_2) & = S(\alpha, M_1) \cup S(\alpha, M_2), \\ S(\alpha, M) & = \emptyset, \textit{ if } \alpha \notin \mathcal{A}(M) \end{cases}$$

For an atom α in a set of messages M, a well-formed selection returns all the atoms in $\mathcal{M}$ if $M = \{\alpha\}$. It returns for it in the union of two sets of messages, the union of the two selections performed in each set separately. It returns the empty set if the atom does not appear in M.

Definition 310 *(Full-invariant-by-intruder selection)*
Let $M \subseteq \mathcal{M}$ such that M is well-protected.
Let $S : \mathcal{A} \times \mathcal{M} \longmapsto 2^{\mathcal{A}}$ be a selection.
We say that S is full-invariant-by-intruder in $\mathcal{C}$ if:
$\forall M \subseteq \mathcal{M}, m \in \mathcal{M}$, *we have:*

$$M \models_{\mathcal{C}} m \Rightarrow \forall \alpha \in \mathcal{A}(m).(S(\alpha, m) \subseteq S(\alpha, M)) \vee (\ulcorner K(I) \urcorner \sqsupseteq \ulcorner \alpha \urcorner)$$

The goal of a full-invariant-by-intruder selection is to create a full-invariant-by-intruder function when composed to an appropriate homomorphism that transforms its returned values into security levels. Since a full-invariant-by-intruder function is requested to resist to any attempt of the intruder to generate a message m from any set of messages M in which the level of security of an atom, that she is not allowed to know, decreases compared to it its value in M, a full-invariant-by-intruder selection is requested to resist to any attempt of the intruder to generate a message m from any set of messages M in which the selection associated to an atom, that she is not allowed to know, could be enlarged compared to the selection associated to this atom in M. This fact is expressed by the definition 310.

Definition 311 *(Reliable selection)*
Let $S : \mathcal{A} \times \mathcal{M} \longmapsto 2^{\mathcal{A}}$ be a selection and $\mathcal{C}$ be a context of verification.

S is $\mathcal{C}$-reliable if S is well-formed and S is full-invariant-by-intruder in $\mathcal{C}$.

3.3.1 Reliable selections inside the protection of an external key

Now, we define a generic class of selections that we call S^{EK}_{Gen} and we prove that any instance of this class is $\mathcal{C}$-reliable. Then we instantiate concrete selections from this class.

Definition 312 (S^{EK}_{Gen}: selection inside the protection of an external key)
We denote by S^{EK}_{Gen} the class of all selections S that meet the following conditions:

- $S(\alpha, \alpha) = \mathcal{A}$; (3.3.1)
- $S(\alpha, m) = \emptyset$, *if* $\alpha \notin \mathcal{A}(m)$; (3.3.2)
- $\forall \alpha \in \mathcal{A}(m)$, *where* $m = f_k(m_1, ..., m_n)$:

$$S(\alpha, m) \subseteq (\bigcup_{1 \leq i \leq n} \mathcal{A}(m_i) \cup \{k^{-1}\} \backslash \{\alpha\}) \text{ if } f_k \in \mathcal{E}_{\mathcal{C}} \text{ and } \ulcorner k^{-1} \urcorner \sqsupseteq \ulcorner \alpha \urcorner \text{ and } m = m_{\Downarrow} \quad (3.3.3)$$

- $\forall \alpha \in \mathcal{A}(m)$, *where* $m = f(m_1, ..., m_n)$:

$$S(\alpha, m) = \begin{cases} \bigcup_{1 \leq i \leq n} S(\alpha, m_i) & \text{if } f_k \in \mathcal{E}_{\mathcal{C}} \text{ and } \ulcorner k^{-1} \urcorner \not\sqsupseteq \ulcorner \alpha \urcorner \text{ and } m = m_{\Downarrow} \quad (a) \\ \bigcup_{1 \leq i \leq n} S(\alpha, m_i) & \text{if } f \in \overline{\mathcal{E}}_{\mathcal{C}} \text{ and } m = m_{\Downarrow} \quad (b) \\ S(\alpha, m_{\Downarrow}) & \text{if } m \neq m_{\Downarrow} \ (c) \end{cases} \quad (3.3.4)$$

- $S(\alpha, \{m\} \cup M) = S(\alpha, m) \cup S(\alpha, M)$ (3.3.5)

For an atom α in an encrypted message $m = f_k(m_1, ..., m_n)$, a selection S as defined above returns a subset (see "$\subseteq$" in equation 3.3.3) among atoms that are neighbors of α in m inside the protection of the most external protective key k including its reverse form k^{-1}. The atom α itself is not selected. This set of candidate atoms is denoted by $\bigcup_{1 \leq i \leq n} \mathcal{A}(m_i) \cup \{k^{-1}\} \setminus \{\alpha\}$ in the equation 3.3.3. The most external protective key (or simply the external key) is the most external one that satisfies $\ulcorner k^{-1} \urcorner \sqsupseteq \ulcorner \alpha \urcorner$. By neighbor of α in m, we mean any atom that travels with it inside the protection of the external key. Outside the protection of the external key, we have:

- no effective selection is performed;
- any two messages joined by an operation other than an encryption by the external key (e.g. a concatenation or an encryption by a weak key) are assimilated to two distinct messages and the selection returns the union of the two selections performed separately in each one (see the equations 3.3.4(a) and 3.3.4(b));
- the selection in two sets of messages is the union of the two selections performed separately in each one(see the equation 3.3.5);
- the selection relative to a clear atom returns all atoms in $\mathcal{M}$ (see the equation 3.3.1);
- the selection relative to an atom that does not appear in a message returns the empty set(see the equation 3.3.2).

S^{EK}_{Gen} defines a generic class of selections since it does not identify what atoms to select precisely inside the protection of the external key. It identifies only the atoms that are candidates for selection and among them we are allowed to return any subset.

Proposition 313
Let $S \in S^{EK}_{Gen}$ and $\mathcal{C}$ be a context of verification.
Let's have a rewriting system $\rightarrow_\xi$ such that $\forall m \in \mathcal{M}, \forall \alpha \in \mathcal{A}(m) \wedge \alpha \notin Clear(m)$, we have:

$$\forall l \rightarrow r \in \rightarrow_\xi, S(\alpha, r) \subseteq S(\alpha, l) \tag{3.3.6}$$

We have:

$$S \text{ is full-invariant-by-intruder in } \mathcal{C}.$$

Proof. See the proof 107 in [16] ■

Remark 314 *(Scope)*

The condition on the rewriting system $\rightarrow_\xi$ given by the equation 3.3.6 in the definition 313 is introduced to make sure that the selection in the normal form is the smallest among all forms of a given message. This prevents the selection S to select atoms that may be inserted maliciously by the intruder by manipulating the equational theory. Hence, we are sure that all selected atoms by S are honest and do not come by an intruder manipulation of the message. For example, let $m = \{\alpha.C\}_{k_{ab}}$ be a message in a homomorphic cryptography (i.e. $\{\alpha.C\}_{k_{ab}} = \{\alpha\}_{k_{ab}}.\{C\}_{k_{ab}}$). In the form $\{\alpha.C\}_{k_{ab}}$, the selection $S(\alpha, \{\alpha.C\}_{k_{ab}})$ may select C, but in the form $\{\alpha\}_{k_{ab}}.\{C\}_{k_{ab}}$, the selection $S(\alpha, \{\alpha\}_{k_{ab}}.\{C\}_{k_{ab}})$ may not. Then, we must make sure that the rewriting system $\rightarrow_\xi$ we are using is oriented in such way that it chooses the form $\{\alpha\}_{k_{ab}}.\{C\}_{k_{ab}}$ rather than the form $\{\alpha.C\}_{k_{ab}}$ because there is no guarantee that C is a honest neighbor and that it had not been inserted maliciously by the intruder using the homomorphic property in the theory. We assume that the equational theory in the context of verification allows the extraction of a convergent rewriting system that meets this condition.

As for the proposition 313, it is quite easy to verify that by constuction a selection S that is instance of S^{EK}_{Gen} is well-formed. The proof of full-invariance-by-intruder is run by induction on the tree of construction of any message. The main idea of the proof is that the selection of the candidate atoms is performed inside the encryption by the most external protective key and therfore beyond the knowledge of the intruder, thus the intruder cannot alter when

she does not detain this key. Besides, according to the lemma 38, in a set of well-protected messages the intruder can never infer this key since it is atomic. The intruder cannot neither use the equational theory to alter this selection for the reasons given in the remark 314. Therefore the set of atoms returned by S cannot be altered (enlarged) by the intruder as required by a full-invariant-by-intruder selection.

Example 315 *Let α be an atomic message and m a message such that $\ulcorner\alpha\urcorner = \{A, B\}$ and $m = \{A.C.\alpha.D\}_{k_{ab}}$. Let S_1, S_2 and S_3 be three selections such that: $S_1(\alpha, m) = \{k_{ab}^{-1}\}$, $S_2(\alpha, m) = \{A, C, k_{ab}^{-1}\}$ and $S_3(\alpha, m) = \{A, C, D, k_{ab}^{-1}\}$. These three selections are $\mathcal{C}$-reliable.*

3.4 Instantiation of reliable selections from the class S_{Gen}^{EK}

Now that we defined a generic class of reliable selections S_{Gen}^{EK}, we will instantiate some concrete selections from it, that are naturally reliable. Instantiating S_{Gen}^{EK} consists in defining selections that return precise sets of atoms among the candidates allowed by S_{Gen}^{EK}.

3.4.1 The selection S_{MAX}^{EK} : is the instance of the class S_{Gen}^{EK} that returns for an atom in a message m all its neighbors, that are principal identities, inside the protection of the external protective key k in addition to its reverse form k^{-1}. (MAX means: the MAXimum of principal identities)

3.4.2 The selection S_{EK}^{EK} : is the instance of the class S_{Gen}^{EK} that returns for an atom in a message m only the reverse form of the external protective key. (EK means: External Key)

3.4.3 The selection S_{N}^{EK} : is the instance of the class S_{Gen}^{EK} that returns only its neighbors, that are principal identities, inside the protection of the external protective key. (N means: Neighbors)

Several other selections could be defined such that the selections inside the most internal key or inside all the keys together since they are all subselections inside the external key.

Example 316
Let α be an atom and m a message such that: $\ulcorner\alpha\urcorner = \{A, C\}$ and $m = \{\{\{\alpha.C\}_{k_{ab}}.B\}_{k_{ac}}.D\}_{k_{ad}}$
$S_{MAX}^{EK}(\alpha, m) = \{C, B, k_{ac}^{-1}\}$; $S_{EK}^{EK}(\alpha, m) = \{k_{ac}^{-1}\}$; $S_{N}^{EK}(\alpha, m) = \{C, B\}$

3.5 Specialized $\mathcal{C}$-reliable selection-based interpretation functions

The following proposition gives the way to transform the elements returned by a selection to security levels.

Proposition 317
Let ψ be a homomorphism defined as follows:

$$\begin{array}{rcl} \psi : (2^{\mathcal{A}})^{\subseteq} & \mapsto & \mathcal{L}^{\sqsupseteq} \\ M & \mapsto & \begin{cases} \top & \text{if } M = \emptyset \\ \bigsqcap\limits_{\alpha \in M} \psi(\alpha) & \text{if not.} \end{cases} \end{array}$$

$$\text{such that: } \psi(\alpha) = \begin{cases} \{\alpha\} \text{ if } \alpha \in \mathcal{I} \text{ (Principal Identities)} \\ \ulcorner\alpha\urcorner \text{ if not.} \end{cases}$$

We have: $F_{MAX}^{EK} = \psi \circ S_{MAX}^{EK}$, $F_{EK}^{EK} = \psi \circ S_{EK}^{EK}$ and $F_{N}^{EK} = \psi \circ S_{N}^{EK}$ are $\mathcal{C}$-reliable.

Proof. See the proof 1110 in [16] ■

The homomorphism ψ defined in the proposition 317 returns for a principal in a selection, its identity. It returns for a key its level of security in the context of verification. ψ ensures the mapping from the operator "$\subseteq$" to the operator "$\sqsupseteq$" in the lattice which enables an interpretation function to inherit the full-invariance-by-intruder from its associated selection. It ensures also the mapping from the operator "$\cup$" to the operator "$\sqcap$" in the lattice, which enables an interpretation function to be well-formed when its associated selection is well-formed. Generally, any function $\psi \circ S$ remains reliable for any selection S that is an instance of S^{EK}_{Gen}.

Example 318
Let α be an atom, m a message and k_{ab} a key such that: $\ulcorner \alpha \urcorner = \{A, B\}$; $m = \{A.C.\alpha.D\}_{k_{ab}}$; $k^{-1}_{ab} = k_{ab}$; $\ulcorner k_{ab} \urcorner = \{A, B\}$;
$S^{EK}_{EK}(\alpha, m) = \{k^{-1}_{ab}\}$; $S^{EK}_{N}(\alpha, m) = \{A, C, D\}$; $S^{EK}_{MAX}(\alpha, m) = \{A, C, D, k^{-1}_{ab}\}$;
$F^{EK}_{EK}(\alpha, m) = \psi \circ S^{EK}_{EK}(\alpha, m) = \ulcorner k^{-1}_{ab} \urcorner = \{A, B\}$; $F^{EK}_{N}(\alpha, m) = \psi \circ S^{EK}_{N}(\alpha, m) = \{A, C, D\}$;
$F^{EK}_{MAX}(\alpha, m) = \psi \circ S^{EK}_{MAX}(\alpha, m) = \{A, C, D\} \sqcap \ulcorner k^{-1} \urcorner = \{A, C, D\} \sqcap \{A, B\} = \{A, C, D, B\}$.

4 The Witness-Functions

In the previous section, we presented a constructive way of a class of selection-based functions that have the required properties to analyze protocols statically. Unfortunatly, they operate only on valid traces that contain closed messages. However, a static protocol analysis should run over a finite set of messages of the generalized roles of the protocol because the set of valid traces is infinite. The finite set of the generalized roles contains variables. The functions we defined are not "enough prepared" to analyze messages with variables because they are not supposed to be full-invariant by substitution (or stable by substitution) [21,22,23]. The full-invariance by substitution is the property-bridge that allows us to perform an analysis over messages with variables and propagate the conclusion made-on to closed messages. In the following section, we deal with the substitution question. We introduce the notion of derivation to reduce the impact of variables and we build the witness-functions that operate on derivative messages rather than messages themselves. As we will see, the witness-functions provide two interesting bounds that are independent of all substitutions and hence we solve the problem of substitution. We last define a criterion of protocol correctness based on these two bounds. The fact that the criterion is independent of all substitutions enables an analysis to be run on generalized roles and the decision made-on to be exported to valid traces.

4.1 Derivative message

Let $m, m_1, m_2 \in \mathcal{M}$; $\mathcal{X}_m = Var(m)$; $S_1, S_2 \subseteq 2^{\mathcal{X}_m}$; $\alpha \in \mathcal{A}(m)$; $X, Y \in \mathcal{X}_m$ and ϵ be the empty message.

Definition 41 *(Derivation)*
We define the derivative message as follows:

$$
\begin{array}{ll}
\partial_X \epsilon & = \epsilon \\
\partial_X \alpha & = \alpha \\
\partial_X X & = \epsilon \\
\partial_X Y & = Y \\
\partial_X f(m) & = f(\partial_X m), f \in \mathcal{E}_{\mathcal{C}} \cup \overline{\mathcal{E}}_{\mathcal{C}} \\
\partial_{\{X\}} m & = \partial_X m \\
\partial_{[X]} m & = \partial_{\{\mathcal{X}_m \setminus X\}} m \\
\partial_{S_1 \cup S_2} m & = \partial_{S_2 \cup S_1} m = \partial_{S_1} \partial_{S_2} m = \partial_{S_2} \partial_{S_1} m
\end{array}
$$

For the sake of simplification, we denote by ∂m the expression $\partial_{\mathcal{X}_m} m$. The operation of derivation in the definition 41 (denoted by ∂) is used to eliminate variables in a message.

$\partial_X m$ consists in eliminating the variable X in m. $\partial[\overline{X}]m$ consists in eliminating all variables, except X, in m. Hence, X when overlined is considered as a constant in m. ∂m consists in eliminating all the variables in m.

Example 42 *Let $m = \{A.X.Y\}_{k_{ab}}$ where A and k_{ab} are static and X and Y are two variables. We have: $\partial m = \{A\}_{k_{ab}}$; $\partial_X m = \{A.Y\}_{k_{ab}}$; $\partial[\overline{X}]m = \{A.X\}_{k_{ab}}$*

Definition 43
Let $m \in \mathcal{M}_p^{\mathcal{G}}$, $X \in \mathcal{X}_m$ and $m\sigma$ be a closed message.
For all $\alpha \in \mathcal{A}(m\sigma)$, $\sigma \in \Gamma$, we denote by:

$$F(\alpha, \partial[\overline{\alpha}]m\sigma) = \begin{cases} \top & \text{if } \alpha \notin \mathcal{A}(m\sigma), \\ F(\alpha, \partial m) & \text{if } \alpha \in \mathcal{A}(\partial m), \\ F(X, \partial[\overline{X}]m) & \text{if } \alpha \in \mathcal{A}(X\sigma) \wedge \alpha \notin \mathcal{A}(\partial m). \end{cases}$$

A message m in a generalized role is composed of two parts: a static part and a dynamic part. The dynamic part is described by variables. For an atom α in the static part (i.e. ∂m), $F(\alpha, \partial[\overline{\alpha}]m\sigma)$ removes the variables in m and gives it the value $F(\alpha, \partial m)$. For anything that is not an atom of the static part, so comes by substitution of some variable X in m, $F(\alpha, \partial[\overline{\alpha}]m\sigma)$ considers it as the variable itself, treated as a constant (as a block), and gives it the value $F(X, \partial[\overline{X}]m)$. It gives the top value for any atom that does not appear in $m\sigma$. For any F such that its associated selection is an instance of the class S_{Gen}^{EK}, $F(\alpha, \partial[\overline{\alpha}]m\sigma)$ depends only on the static part of m since α is never selected. The introduction of the derivation could suggest that we give to $F(\alpha, m\sigma)$ the value of $F(\alpha, \partial[\overline{\alpha}]m\sigma)$ and hence we neutralize the variable effects. Unfortunately, this does not happen without undesirable "side-effects" because derivation causes a "loss of details". Let's examine the following case in the example 44.

Example 44
Let m_1 and m_2 be two messages of a generalized role of a protocol p such that $m_1 = \{\alpha.B.X\}_{k_{ad}}$ and $m_2 = \{\alpha.Y.C\}_{k_{ad}}$ and $\ulcorner\alpha\urcorner = \{A, D\}$. Let $m = \{\alpha.B.C\}_{k_{ad}}$ be in a valid trace generated by p.

$$F_{MAX}^{EK}(\alpha, \partial[\overline{\alpha}]m) = \begin{cases} \{B, A, D\} & \text{if } m \text{ comes by the substitution of } X \text{ by } C \text{ in } m_1 \\ \{A, D, C\} & \text{if } m \text{ comes by the substitution of } Y \text{ by } B \text{ in } m_2 \end{cases}$$

Hence $F_{MAX}^{EK}(\alpha, \partial[\overline{\alpha}]m)$ is not even a function on the closed message m since it may return more than one image for the same preimage. This leads us straightly to the witness-functions.

4.2 The Witness-Functions

Definition 45 *(Witness-Function)*
Let $m \in \mathcal{M}_p^{\mathcal{G}}$, $X \in \mathcal{X}_m$ and $m\sigma$ be a closed message.
Let p be a protocol and F be a $\mathcal{C}$-reliable interpretation function.
We define a witness-function $\mathcal{W}_{p,F}$ for all $\alpha \in \mathcal{A}(m\sigma)$, $\sigma \in \Gamma$, as follows:

$$\mathcal{W}_{p,F}(\alpha, m\sigma) = \bigsqcap_{\substack{m' \in \mathcal{M}_p^{\mathcal{G}} \\ \exists \sigma' \in \Gamma. m'\sigma' = m\sigma}} F(\alpha, \partial[\overline{\alpha}]m'\sigma')$$

$\mathcal{W}_{p,F}$ is said to be a witness-function inside the protection of an external key when F is an interpretation function such that its associated selection is an instance of the class S_{Gen}^{EK}.

As seen in the example 44, the application defined in 43 is not necessary a function in the set $\mathcal{M}_p^{\mathcal{G}}$ of messages generated by the generalized roles of p since a valid trace could have more than one source in $\mathcal{M}_p^{\mathcal{G}}$ such that each source has a different static part. A witness-function

is though a function since it looks for all the sources of any closed message and takes the minimum (the union). This minimum exists and is unique in the finite set $\mathcal{M}_p^{\mathcal{G}}$. Although a witness-function is protocol-dependent since it depends on messages in the generalized roles of the protocol, it is built in a standard way for any pair (protocol, interpretation function) in input.

Remark 46 *For a witness-function inside the protection of an external key, since its associated interpretation function calculates the security level of an atom always in a message m having an encryption pattern, i.e. when $f_k \in \mathcal{E}_{\mathcal{C}}$, the search of all sources of m in the set $\{m' \in \mathcal{M}_p^{\mathcal{G}} | \exists \sigma' \in \Gamma . m'\sigma' = m\sigma\}$ is reduced to a search in the encryption patterns in $\mathcal{M}_p^{\mathcal{G}}$.*

4.3 Inheritance of reliability properties from F

Proposition 47
Let $\mathcal{W}_{p,F}$ be a witness-function inside the protection of an external key.
We have:

$$\mathcal{W}_{p,F} \text{ inherits reliability from } F$$

Proof. See the proof 122 in [16] ■

The selection associated with a witness-function inside the protection of an external key is the union of selections associated with the interpretation function F restricted to derivative messages. It is easy to verify that a witness-function is by construction well-formed. For the full-invariance-by-intruder property, since the derivation just removes variables and since each selection in the union returns a subset among acceptable candidates, then the union itself returns a subset among acceptable candidates (the union of subsets is a subset). Therefore the selection associated with a witness-function remains an instance of the class S_{Gen}^{EK} and so full-invariant-by-intruder. Since the witness-function is the composition of the homomorphism associated with F and an instance of the class S_{Gen}^{EK}, then it is reliable.

Example 48
Let $\mathcal{M}_p^{\mathcal{G}} = \{\{\alpha.B.X\}_{k_{ad}}, \{\alpha.Y.C\}_{k_{ad}}, \{A.Z\}_{k_{bc}}\}$; $m_1 = \{\alpha.B.C\}_{k_{ad}}$; $Var(\mathcal{M}_p^{\mathcal{G}}) = \{X, Y, Z\}$.

$\mathcal{W}_{p,F_{MAX}^{EK}}(\alpha, m_1)$
$=$ *{Definition 45}*
$\underset{\substack{m' \in \mathcal{M}_p^{\mathcal{G}} \\ \exists \sigma' \in \Gamma . m'\sigma' = m_1}}{\sqcap} F_{MAX}^{EK}(\alpha, \partial[\overline{\alpha}]m'\sigma') = \underset{\substack{\{\{\alpha.B.X\}_{k_{ad}}, \{\alpha.Y.C\}_{k_{ad}}\} \\ \sigma' = \{X \longmapsto C, Y \longmapsto B\}}}{\sqcap} F_{MAX}^{EK}(\alpha, \partial[\overline{\alpha}]m'\sigma')$
$=$ *{$\mathcal{W}_{p,F_{MAX}^{EK}}$ is well-formed from the proposition 47 }*
$F_{MAX}^{EK}(\alpha, \partial[\overline{\alpha}]\{\alpha.B.X\}_{k_{ad}}[X \longmapsto C]) \sqcap F_{MAX}^{EK}(\alpha, \partial[\overline{\alpha}]\{\alpha.Y.C\}_{k_{ad}}[Y \longmapsto B])$
$=$*{Definition 43 and derivation in 41}*
$F_{MAX}^{EK}(\alpha, \{\alpha.B\}_{k_{ad}}) \sqcap F_{MAX}^{EK}(\alpha, \{\alpha.C\}_{k_{ad}})$
$=$ *{Definition of F_{MAX}^{EK}}*
$\{B, A, D\} \cup \{C, A, D\} = \{B, A, D, C\}$

4.4 Bounding a Witness-Function

Lemma 49
Let $m \in \mathcal{M}_p^{\mathcal{G}}$ and $\mathcal{W}_{p,F}$ be a witness-function inside the protection of an external key.
$\forall \sigma \in \Gamma, \forall \alpha \in \mathcal{A}(\mathcal{M}_p)$ we have:

$$F(\alpha, \partial[\overline{\alpha}]m) \sqsupseteq \mathcal{W}_{p,F}(\alpha, m\sigma) \sqsupseteq \underset{\substack{m' \in \mathcal{M}_p^{\mathcal{G}} \\ \exists \sigma' = mgu(m', m)}}{\sqcap} F(\alpha, \partial[\overline{\alpha}]m'\sigma')$$

Proof. See the proof 124 in [16] ■

The upper bound of a witness-function estimates the security level of an atom from one *confirmed* source of the closed message, the witness-function it-self estimates it from the *exact* sources of the closed message (i.e. when the protocol is executed), and the lower bound estimates it from all *likely* sources of the closed message. The unification in the lower bound "hunts" the candidates from all the *likely* sources of the closed message in the protocol.

4.5 Sufficient condition for protocol correctness with a Witness-Function

Now, it is time to state the protocol analysis with a Witness-Function theorem that states a criterion for protocol correctness with respect to the secrecy property which is independent of all substitutions.

Theorem 41 *(Protocol analysis with a Witness-Function)*
Let $\mathcal{W}_{p,F}$ be a witness-function inside the protection of an external key.
A sufficient condition of correctness of p with respect to the secrecy property is:
$\forall R.r \in R_G(p), \forall \alpha \in \mathcal{A}(r^+)$ *we have:*

$$\sqcap_{\substack{m' \in \mathcal{M}_p^{\mathcal{G}} \\ \exists \sigma' = mgu(m', r^+)}} F(\alpha, \partial[\overline{\alpha}]m'\sigma') \sqsupseteq \ulcorner \alpha \urcorner \sqcap F(\alpha, \partial[\overline{\alpha}]R^-)$$

Proof. See the proof 125 in [16] ■

Thanks to the independence of the criterion stated by the theorem 41 of all substitutions, any decision made on the generalized roles can be transmitted to valid traces.

5 NSL protocol analysis with a witness-function

Hereafter, we analyze the NSL protocol given in Table 1 with a witness-function.

Let's have a context of verification such that: $\ulcorner A \urcorner = \bot$; $\ulcorner B \urcorner = \bot$; $\ulcorner N_a^i \urcorner = \{A, B\}$ (secret shared between A and B); $\ulcorner N_b^i \urcorner = \{A, B\}$ (secret shared between A and B); $\ulcorner k_a^{-1} \urcorner = \{A\}$; $\ulcorner k_b^{-1} \urcorner = \{B\}$; $(\mathcal{L}, \sqsupseteq, \sqcup, \sqcap, \bot, \top) = (2^{\mathcal{I}}, \subseteq, \cap, \cup, \mathcal{I}, \emptyset)$; $\mathcal{I} = \{I, A, B, A_1, A_2, B_1, B_2, ...\}$;
The set of messages generated by the protocol is $\mathcal{M}_p^{\mathcal{G}} = \{\{N_{A_1}.A_1\}_{k_{B_1}}, \{B_2.N_{A_2}\}_{k_{A_2}}, \{B_3.X_1\}_{k_{A_3}}, \{X_2\}_{k_{B_4}}, \{Y_1.A_4\}_{k_{B_5}}, \{B_6.Y_2\}_{k_{A_5}}, \{B_7.N_{B_7}\}_{k_{A_6}}, \{N_{B_8}\}_{k_{B_8}}\}$
The variables are denoted by X_1, X_2, Y_1 and Y_2;
The static names are denoted by N_{A_1}, A_1, k_{B_1}, B_2, N_{A_2}, k_{A_2}, B_3, k_{A_3}, k_{B_4}, A_4, k_{B_5}, B_6, k_{A_5}, B_7, N_{B_7}, k_{A_6}, N_{B_8} and k_{B_8}.

After elimination of duplicates, $\mathcal{M}_p^{\mathcal{G}} = \{\{N_{A_1}.A_1\}_{k_{B_1}}, \{B_2.N_{A_2}\}_{k_{A_2}}, \{B_3.X_1\}_{k_{A_3}}, \{X_2\}_{k_{B_4}}, \{Y_1.A_4\}_{k_{B_5}}, \{B_7.N_{B_7}\}_{k_{A_6}}, \{N_{B_8}\}_{k_{B_8}}\}$
Let's select the Witness-Function as follows:
$p = NSL$; $F = F_{MAX}^{EK}$; $\mathcal{W}_{p,F}(\alpha, m\sigma) = \sqcap_{\substack{m' \in \mathcal{M}_p^{\mathcal{G}} \\ \exists \sigma'. m'\sigma' = m\sigma}} F(\alpha, \partial[\overline{\alpha}]m'\sigma')$;

Let's denote the lower bound of the Witness-Function in the theorem 41 by:

$$\mathcal{W}'_{p,F}(\alpha, r^+) = \sqcap_{\substack{m' \in \mathcal{M}_p^{\mathcal{G}} \\ \exists \sigma' = mgu(m', r^+)}} F(\alpha, \partial[\overline{\alpha}]m'\sigma')$$

Please notice that all messages are in their normal form since no equational theory is defined. Principal identities in the context are not analyzed since they are declared public. The protocol is analyzed for the property of secrecy only.

5.1 Analysis of the generalized role of A

As defined in the generalized roles of p, an agent A can participate in two consequent sessions: S^i and S^j such that $j > i$. In the former session S^i, the agent A receives nothing and sends the message $\{N_a^i.A\}_{k_b}$. In the consequent session S^j, she receives the message $\{B.N_a^i\}_{k_a}.\{B.X\}_{k_a}$ and she sends the message $A.B.\{X\}_{k_b}$. This is described by the following schema:

$$S^i : \frac{\square}{\{N_a^i.A\}_{k_b}} \qquad\qquad S^j : \frac{\{B.N_a^i\}_{k_a}.\{B.X\}_{k_a}}{A.B.\{X\}_{k_b}}$$

-Analysis of the messages exchanged in the session S^i:

1- For any N_a^i:
a- When sending: $r^+_{S^i} = \{N_a^i.A\}_{k_b}$ *(in a sending step, the lower bound is used)*
$\forall N_a^i.\{m' \in \mathcal{M}_p^{\mathcal{G}} | \exists \sigma' = mgu(m', r^+_{S^i})\}$
$= \forall N_a^i.\{m' \in \mathcal{M}_p^{\mathcal{G}} | \exists \sigma' = mgu(m', \{N_a^i.A\}_{k_b})\}$
$= \{(\{N_{A_1}.A_1\}_{k_{B_1}}, \sigma'_1), (\{X_2\}_{k_{B_4}}, \sigma'_2), (\{Y_1.A_4\}_{k_{B_5}}, \sigma'_3)\}$ such that:

$$\begin{cases} \sigma'_1 = \{N_{A_1} \longmapsto N_a^i, A_1 \longmapsto A, k_{B_1} \longmapsto k_b\} \\ \sigma'_2 = \{X_2 \longmapsto N_a^i.A, k_{B_4} \longmapsto k_b\} \\ \sigma'_3 = \{Y_1 \longmapsto N_a^i, A_4 \longmapsto A, k_{B_5} \longmapsto k_b\} \end{cases}$$

$\mathcal{W}'_{p,F}(N_a^i, \{N_a^i.A\}_{k_b})$
$=$ {Definition of the lower bound of the Witness-Function}
$F(N_a^i, \partial[\overline{N_a^i}]\{N_{A_1}.A_1\}_{k_{B_1}}\sigma'_1) \sqcap F(N_a^i, \partial[\overline{N_a^i}]\{X_2\}_{k_{B_4}}\sigma'_2) \sqcap F(N_a^i, \partial[\overline{N_a^i}]\{Y_1.A_4\}_{k_{B_5}}\sigma'_3)$
$=$ {Setting the static neighborhood}
$F(N_a^i, \partial[\overline{N_a^i}]\{N_a^i.A\}_{k_b}\sigma'_1) \sqcap F(N_a^i, \partial[\overline{N_a^i}]\{X_2\}_{k_b}\sigma'_2) \sqcap F(N_a^i, \partial[\overline{N_a^i}]\{Y_1.A\}_{k_b}\sigma'_3)$
$=$ {Definition 43}
$F(N_a^i, \{N_a^i.A\}_{k_b}) \sqcap F(X_2, \partial[\overline{X_2}]\{X_2\}_{k_b}) \sqcap F(Y_1, \partial[\overline{Y_1}]\{Y_1.A\}_{k_b})$
$=$ {Derivation in the definition 41}
$F(N_a^i, \{N_a^i.A\}_{k_b}) \sqcap F(X_2, \{X_2\}_{k_b}) \sqcap F(Y_1, \{Y_1.A\}_{k_b})$
$=$ {Since $F = F^{EK}_{MAX}$}
$\{A, B\} \cup \{B\} \cup \{A, B\} = \{A, B\}$(1.0)

b- When receiving: $R^-_{S^i} = \emptyset$ *(in a receiving step, the upper bound is used)*
$F(N_a^i, \partial[\overline{N_a^i}]\emptyset) = F(N_a^i, \emptyset) = \top$ (1.1)

2- Compliance with the correctness criterion stated in the theorem 41:
From (1.0) and (1.1), we have: $\mathcal{W}'_{p,F}(N_a^i, \{N_a^i.A\}_{k_b}) = \{A, B\} \sqsupseteq \ulcorner N_a^i \urcorner \sqcap F(N_a^i, \emptyset) = \{A, B\}$ (1.2)
From (1.2) we have: the messages exchanged in the session S^i respect the correctness criterion stated in the theorem 41. **(I)**

-Analysis of the messages exchanged in the session S^j:

1- For any X:
a- When sending: $r^+_{S^j} = A.B.\{X\}_{k_b}$
$\mathcal{W}'_{p,F}(X, A.B.\{X\}_{k_b}) = \mathcal{W}'_{p,F}(X, A) \sqcap \mathcal{W}'_{p,F}(X, B) \sqcap \mathcal{W}'_{p,F}(X, \{X\}_{k_b}) = \top \sqcap \top \sqcap \mathcal{W}'_{p,F}(X, \{X\}_{k_b}) = \mathcal{W}'_{p,F}(X, \{X\}_{k_b})$ (2.0)
$\forall X.\{m' \in \mathcal{M}_p^{\mathcal{G}} | \exists \sigma' = mgu(m', \{X\}_{k_b})\} = \{(\{X_2\}_{k_{B_4}}, \sigma'_1)\}$ such that:

$$\sigma'_1 = \{X_2 \longmapsto X, k_{B_4} \longmapsto k_b\}$$

$\mathcal{W}'_{p,F}(X, \{X\}_{k_b})$
$=$ {Definition of the lower bound of the Witness-Function}

$F(X, \partial[\overline{X}]\{X_2\}_{k_{B_4}}\sigma'_1)$
= {Setting the static neighborhood}
$F(X, \partial[\overline{X}]\{X_2\}_{k_b}\sigma'_1)$
= {Definition 43}
$F(X_2, \partial[\overline{X_2}]\{X_2\}_{k_b})$
= {Derivation in the definition 41}
$F(X_2, \{X_2\}_{k_b})$
= {Since $F = F^{EK}_{MAX}$}
$\{B\}$ (2.1)

b- When receiving: $R^-_{S^j} = \{B.N^i_a\}_{k_a}.\{B.X\}_{k_a}$ *(in a receiving step, the upper bound is used)*
$F(X, \partial[\overline{X}]\{B.N^i_a\}_{k_a}.\{B.X\}_{k_a}) = F(X, \partial[\overline{X}]\{B.N^i_a\}_{k_a}) \sqcap F(X, \partial[\overline{X}]\{B.X\}_{k_a}) =$
$F(X, \{B.N^i_a\}_{k_a}) \sqcap F(X, \{B.X\}_{k_a}) = \top \sqcap \{A, B\} = \{A, B\}$ (2.2)

3-Compliance with the correctness criterion stated in the theorem 41:
From (2.0), (2.1) and (2.2), we have:
$\mathcal{W}'_{p,F}(X, A.B.\{X\}_{k_b}) = \{B\} \sqsupseteq \ulcorner X \urcorner \sqcap F(X, \partial[\overline{X}]\{B.N^i_a\}_{k_a}.\{B.X\}_{k_a}) = \ulcorner X \urcorner \cup \{A, B\}$ (2.3)
From (2.3), we have: the messages exchanged in the session S^j respect the correctness criterion stated in the theorem 41. (II)
From (I) and (II), the messages exchanged in the generalized role of A respect the correctness criterion stated in the theorem 41. (III)

5.2 Analysis of the generalized role of B

As defined in the generalized roles of p, an agent B can participate in a session S'^i, in which she receives the message $\{Y.A\}_{k_b}$ and she sends the message $\{B.Y\}_{k_a}.\{B.N^i_b\}_{k_a}$. This is described by the following schema:

$$S'^i : \frac{\{Y.A\}_{k_b}}{\{B.Y\}_{k_a}.\{B.N^i_b\}_{k_a}}$$

1- For any N^i_b:
a- When sending: $r^+_{S'^i} = \{B.Y\}_{k_a}.\{B.N^i_b\}_{k_a}$ *(in a sending step, the lower bound is used)*
$\mathcal{W}'_{p,F}(N^i_b, \{B.Y\}_{k_a}.\{B.N^i_b\}_{k_a}) = \mathcal{W}'_{p,F}(N^i_b, \{B.Y\}_{k_a}) \sqcap \mathcal{W}'_{p,F}(N^i_b, \{B.N^i_b\}_{k_a}) =$
$\top \sqcap \mathcal{W}'_{p,F}(N^i_b, \{B.N^i_b\}_{k_a}) = \mathcal{W}'_{p,F}(N^i_b, \{B.N^i_b\}_{k_a})$ (3.0)
$\forall N^i_b.\{m' \in \mathcal{M}^{\mathcal{G}}_p | \exists \sigma' = mgu(m', \{B.N^i_b\}_{k_a})\}$
= $\{(\{B_3.X_1\}_{k_{A_3}}, \sigma'_1), (\{X_2\}_{k_{B_4}}, \sigma'_2), (\{B_7.N_{B_7}\}_{k_{A_6}}, \sigma'_3)\}$ such that:

$$\begin{cases} \sigma'_1 = \{B_3 \longmapsto B, X_1 \longmapsto N^i_b, k_{A_3} \longmapsto k_a\} \\ \sigma'_2 = \{X_2 \longmapsto B.N^i_b, k_{B_4} \longmapsto k_a\} \\ \sigma'_3 = \{B_7 \longmapsto B, N_{B_7} \longmapsto N^i_b, k_{A_6} \longmapsto k_a\} \end{cases}$$

$\mathcal{W}'_{p,F}(N^i_b, \{B.N^i_b\}_{k_a})$
= {Definition of the lower bound of the Witness-Function}
$F(N^i_b, \partial[\overline{N^i_b}]\{B_3.X_1\}_{k_{A_3}}\sigma'_1) \sqcap F(N^i_b, \partial[\overline{N^i_b}]\{X_2\}_{k_{B_4}}\sigma'_2) \sqcap F(N^i_b, \partial[\overline{N^i_b}]\{B_7.N_{B_7}\}_{k_{A_6}}\sigma'_3)$
= {Setting the static neighborhood}
$F(N^i_b, \partial[\overline{N^i_b}]\{B.X_1\}_{k_a}\sigma'_1) \sqcap F(N^i_b, \partial[\overline{N^i_b}]\{X_2\}_{k_a}\sigma'_2) \sqcap F(N^i_b, \partial[\overline{N^i_b}]\{B.N^i_b\}_{k_a}\sigma'_3)$
= {Definition 43}
$F(X_1, \partial[\overline{X_1}]\{B.X_1\}_{k_a}) \sqcap F(X_2, \partial[\overline{X_2}]\{X_2\}_{k_a}) \sqcap F(N^i_b, \partial[\overline{N^i_b}]\{B.N^i_b\}_{k_a})$
= {Derivation in the definition 41}
$F(X_1, \{B.X_1\}_{k_a}) \sqcap F(X_2, \{X_2\}_{k_a}) \sqcap F(N^i_b, \{B.N^i_b\}_{k_a})$
= {Since $F = F^{EK}_{MAX}$}
$\{A, B\} \cup \{A\} \cup \{A, B\} = \{A, B\}$ (3.1)

b- When receiving: $R^{-}_{S'^{i}} = \{Y.A\}_{k_b}$ *(in a receiving step, the upper bound is used)*
$F(N^i_b, \partial[\overline{N^i_b}]\{Y.A\}_{k_b}) = \top$ (3.2)

2- For any Y:
a- When sending: $r^{+}_{S'^{i}} = \{B.Y\}_{k_a}.\{B.N^i_b\}_{k_a}$ *(in a receiving step, the upper bound is used)*
$\mathcal{W}'_{p,F}(Y, \{B.Y\}_{k_a}.\{B.N^i_b\}_{k_a}) = \mathcal{W}'_{p,F}(Y, \{B.Y\}_{k_a}) \sqcap \mathcal{W}'_{p,F}(Y, \{B.N^i_b\}_{k_a}) =$
$\mathcal{W}'_{p,F}(Y, \{B.Y\}_{k_a}) \sqcap \top = \mathcal{W}'_{p,F}(Y, \{B.Y\}_{k_a})$ (3.3)

$\forall Y.\{m' \in \mathcal{M}^{\mathcal{G}}_p | \exists \sigma' = mgu(m', \{B.Y\}_{k_a})\} = \{(\{B_3.X_1\}_{k_{A_3}}, \sigma_1), (\{X_2\}_{k_{B_4}}, \sigma_2)\}$ such that:

$$\begin{cases} \sigma'_1 = \{B_3 \longmapsto B, X_1 \longmapsto Y, k_{A_3} \longmapsto k_a\} \\ \sigma'_2 = \{X_2 \longmapsto B.Y, k_{B_4} \longmapsto k_a\} \end{cases}$$

$\mathcal{W}'_{p,F}(Y, \{B.Y\}_{k_a})$
= {Definition of the lower bound of the Witness-Function}
$F(Y, \partial[\overline{Y}]\{B_3.X_1\}_{k_{A_3}}\sigma'_1) \sqcap F(Y, \partial[\overline{Y}]\{X_2\}_{k_{B_4}}\sigma'_2)$
= {Setting the static neighborhood}
$F(Y, \partial[\overline{Y}]\{B.X_1\}_{k_a}\sigma'_1) \sqcap F(Y, \partial[\overline{Y}]\{X_2\}_{k_a}\sigma'_2) =$
= {Definition 43}
$F(X_1, \partial[\overline{X_1}]\{B.X_1\}_{k_a}) \sqcap F(X_2, \partial[\overline{X_2}]\{X_2\}_{k_a})$
= {Derivation in the definition 41}
$F(X_1, \{B.X_1\}_{k_a}) \sqcap F(X_2, \{X_2\}_{k_a})$
= {Since $F = F^{EK}_{MAX}$}
$\{A, B\} \cup \{A\} = \{A, B\}$ (3.4)

b- When receiving: $R^{-}_{S'^{i}} = \{Y.A\}_{k_b}$ *(in a receiving step, the upper bound is used)*
$F(Y, \partial[\overline{Y}]\{Y.A\}_{k_b}) = \{A, B\}$ (3.5)

3- Compliance with the correctness criterion stated in the theorem 41:
From (3.0), (3.1) and (3.2) we have:
$\mathcal{W}'_{p,F}(N^i_b, \{B.Y\}_{k_a}.\{B.N^i_b\}_{k_a}) = \{A, B\} \sqsupseteq \ulcorner N^i_b \urcorner \sqcap F(N^i_b, \partial[\overline{N^i_b}]\{Y.A\}_{k_b}) = \{A, B\}$ (3.6)
From (3.3), (3.4) and (3.5) we have:
$\mathcal{W}'_{p,F}(Y, \{B.Y\}_{k_a}.\{B.N^i_b\}_{k_a}) = \{A, B\} \sqsupseteq \ulcorner Y \urcorner \sqcap F(Y, \partial[\overline{Y}]\{Y.A\}_{k_b}) = \ulcorner Y \urcorner \sqcap \{A, B\}$ (3.7)
From (3.6) and (3.7), the messages exchanged in the session S'^{i} respect the correctness criterion stated in the theorem 41. (IV)
From (IV), the messages exchanged in the generalized role of B respect the correctness criterion stated by the theorem 41. (V)
From (III) and (V), the protocol NSL respects the correctness criterion stated by the theorem 41, then it is correct with respect to the secrecy property.

6 Comparison with related works

Houmani et al. in [8,9,10,11] have defined universal functions to analyze increasing protocols. Even if they gave a clear guideline to build safe functions, just two functions have been defined: DEK and DEKAN. That is due to the difficulty to prove that a function is full-invariant by substitution. Henceforth, with a witness-function, there is no need to request the property of full-invariance by substitution from an interpretation function. The witness-function takes it in input, without this property, and solves the problem of substitutions locally in the protocol, thanks to its construction that depends fully on the static part of a message and thanks to its two bounds that do not depend on substitutions (bring two properties and have one for free). This enables us to build more functions, and then, to analyze more protocols. The bounds of a witness-function having an output free of variables

are expected to prove the correctness of protocols that could not be proven with universal functions because of variables in output, for instance, Kerberos protocol and SET protocol with Houmanis' functions. Our approach does not need a complicated formalism as do the rank-functions suggested by Schneider in [1] that require the CSP formalism [24]. It does not need, neither, a strongly message-typing as suggested by Abadi in [5]. Our approach intersects Schneider's work and Houmani's work in the way of seeing the protocol correctness through its growth and through the need to have reliable metrics to evaluate security. Besides, all of these methods handle an unbounded number of sessions and provide a semi-decidable procedure since they decide only when the protocol is increasing.

7 Conclusion and future work

In this paper, we introduced the witness-functions as a technique to analyze cryptographic protocols for secrecy. We gave the way to build them. In a future work, we intend to define witness-functions based on the selection of other keys (other than the most external key) like the most internal key or all encryption keys together. In this respect, we believe that a witness-function based on the selection of all neighbors and all encryption keys (i.e. $S(\alpha, \{\{E.\alpha\}_{k_{AB}}\}_{k_{CD}}) = \{E, k_{AB}^{-1}, k_{CD}^{-1}\}$) could efficiently deal with protocols with the Diffie-Hellman property in a non-empty equational theory. In this respect, we believe that the witness-functions are ready to deal with algebraic properties in convergent theories. In addition, we intend to take benefice of the bounds of a witness-function to consider exchanged messages in a protocol as keys of encryption. We intend also to experiment our witness-functions on compose protocols. Indeed, many protocols could be secure when they are analyzed separately but may show undesirable interactions when analyzed together. In this regard, similar researches had been led by V. Cortier, S. Ciobaca and S. Delaune in [25,26,27] where they suggest protocol-tags to secure compose protocols. We think that our witness-functions are prepared to deal with this question.

References

1. Steve Schneider. Verifying authentication protocols in csp. *IEEE Trans. Software Eng.*, 24(9):741–758, 1998.
2. Steve Schneider. Security properties and csp. In *IEEE Symposium on Security and Privacy*, pages 174–187, 1996.
3. Steve A. Schneider and Rob Delicata. Verifying security protocols: An application of csp. In *25 Years Communicating Sequential Processes*, pages 243–263, 2004.
4. James Heather and Steve Schneider. A decision procedure for the existence of a rank function. *J. Comput. Secur.*, 13(2):317–344, March 2005.
5. Martín Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46:611–638, 1998.
6. Martín Abadi and Andrew D. Gordon. Reasoning about cryptographic protocols in the spi calculus. In *CONCUR*, pages 59–73, 1997.
7. Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *ACM Conference on Computer and Communications Security*, pages 36–47, 1997.
8. Hanane Houmani and Mohamed Mejri. Practical and universal interpretation functions for secrecy. In *SECRYPT*, pages 157–164, 2007.
9. Hanane Houmani and Mohamed Mejri. Ensuring the correctness of cryptographic protocols with respect to secrecy. In *SECRYPT*, pages 184–189, 2008.
10. Hanane Houmani and Mohamed Mejri. Formal analysis of set and nsl protocols using the interpretation functions-based method. *Journal Comp. Netw. and Communic.*, 2012, 2012.
11. Hanane Houmani, Mohamed Mejri, and Hamido Fujita. Secrecy of cryptographic protocols under equational theory. *Knowl.-Based Syst.*, 22(3):160–173, 2009.
12. Mourad Debbabi, Y. Legaré, and Mohamed Mejri. An environment for the specification and analysis of cryptoprotocols. In *ACSAC*, pages 321–332, 1998.

13. Mourad Debbabi, Mohamed Mejri, Nadia Tawbi, and I. Yahmadi. Formal automatic verification of authentication crytographic protocols. In *ICFEM*, pages 50–59, 1997.
14. Mourad Debbabi, Mohamed Mejri, Nadia Tawbi, and I. Yahmadi. From protocol specifications to flaws and attack scenarios: An automatic and formal algorithm. In *WETICE*, pages 256–262, 1997.
15. Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.
16. Jaouhar Fattahi, Mohamed Mejri, and Hanane Houmani. The witness-functions: Proofs and intermediate results `http://web_security.fsg.ulaval.ca/lab/sites/default/files/WF/WFFMS/WFAppend.pdf`. pages 1–28, 2014.
17. Bruno Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 17(4):363–434, 2009.
18. Hubert Comon-Lundh, Véronique Cortier, and Eugen Zalinescu. Deciding security properties for cryptographic protocols. application to key cycles. *ACM Trans. Comput. Log.*, 11(2), 2010.
19. Véronique Cortier and Stéphanie Delaune. Decidability and combination results for two notions of knowledge in security protocols. *J. Autom. Reasoning*, 48(4):441–487, 2012.
20. Véronique Cortier, Steve Kremer, and Bogdan Warinschi. A survey of symbolic methods in computational analysis of cryptographic systems. *J. Autom. Reasoning*, 46(3-4):225–259, 2011.
21. Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.
22. Nachum Dershowitz and David A. Plaisted. Rewriting. In *Handbook of Automated Reasoning*, pages 535–610. 2001.
23. Hubert Comon-Lundh, Claude Kirchner, and Hélène Kirchner, editors. *Rewriting, Computation and Proof, Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, volume 4600 of *Lecture Notes in Computer Science*. Springer, 2007.
24. Steve A. Schneider, Helen Treharne, and Neil Evans. Chunks: Component verification in csp||b. In *IFM*, pages 89–108, 2005.
25. Stefan Ciobaca and Veronique Cortier. Protocol composition for arbitrary primitives. *2012 IEEE 25th Computer Security Foundations Symposium*, 0:322–336, 2010.
26. Véronique Cortier. Secure composition of protocols. In *TOSCA*, pages 29–32, 2011.
27. Véronique Cortier and Stéphanie Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1):1–36, 2009.

Formal Enforcement of Security Policies on Choreographed Services

Mahjoub Langar and Karim Dahmani

LIP2 Research Laboratory, Faculté des Sciences de Tunis, Tunisia

Abstract. Web services are software systems that support distributed applications composed of independent processes which communicate by message passing. To realize the full potential of web services, we need to compose existent web services in order to get more functionalities. However the composition of web services should be secure. In this paper we propose an automatic formal approach to monitor the execution of a choreography of web services and we prove its correctness. We introduce the syntax and semantic rules of a new operator which takes as input a choreography and a security policy and produces as output a secure version of this choreography which behaves like the original one and does not violate the security policy.

Keywords: Choreographed services, web service security, formal verification, runtime verification

1 Introduction

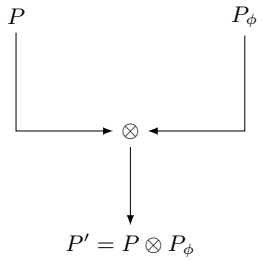
With the explosive growth of the Internet, intranet and electronic commerce, the concept of web services has emerged in the world to be exploited by most cross-organizational boundaries. In fact, web services provide a suitable technical foundation for making business processes accessible within and across enterprises. Moreover web services are software systems that support distributed applications composed of independent processes which communicate by message passing. But for an appropriate exploitation of web services, we need to compose them. Indeed, individual web services often offer limited capabilities. In some situations, a client's request cannot be satisfied by a single web service. However, when composed with other web services, they can satisfy the client demand. To realize the full potential of web services, we need to compose existent web services in order to get more functionalities. Web service composition rules deal with how different services are composed into a coherent global service. In particular, they specify the order in which services are invoked, and the conditions under which a certain service may or may not be invoked. Among the approaches investigated in service composition, we distinguish orchestration and choreography. The orchestration composes available services and adds a central coordinator (the orchestrater) which is responsible for invoking and composing the single sub-activities. However the second one, referred to as web

service choreography, does not assume the exploitation of a central coordinator but it defines complex tasks via the definition of the conversation that should be undertaken by each participant. While several proposals exist for orchestration languages (e.g. Business Process Execution Language (BPEL)[1], Web Services Flow Language (WSFL)[2]), choreography languages are still in a preliminary stage of definition. A proposal, Web Service Choreography Description Language (WS-CDL)[3], was issued by the World Wide Web Consortium (W3C) in December 2004.

The need of secure web service composition has led to a great interest from researchers of the last decade. However most of the works focus on how to guarantee desired properties such as correctness, deadlock avoidance, etc. In our work, in addition to these properties, we introduce an automatic approach to monitor a choreography by specifying and enforcing security policies on web services and we prove its correctness using formal methods. Some works such as [4,5] have enforced security policies on concurrent systems using process algebra. Formal methods are mathematical techniques that are well-suited to address the aforementioned issues. A variety of proposals to formally describe, compose and verify web service compositions using the formal methods exists. In [6], a Petri-net based design and verification framework for web service composition is proposed. In [7], the authors introduce a Petri-net based algebra to compose web services based on control flows. In [8], authors use Milner's process algebra CCS to specify and compose web services as processes, and the Concurrency Workbench to validate properties such as correct web service composition. In [9], the authors provide an encoding of BPEL processes into web service timed state transition systems, a formalism that is closely related to timed automata and discuss a framework in which timed assumptions can be model checked.

In this paper, we propose an enforcement of security policies over web services using formal methods. We present an automated approach for monitoring the behavior of a service in a choreography. More precisely, we define an operator $\otimes$ that takes as input a process P and a security policy P_ϕ and generates a new process $P' = P \otimes P_\phi$ which respects the following conditions :

- $P' \mid\sim P_\phi$, i.e. P' satisfies the security policy P_ϕ.
- $P' \sqsubseteq P$, i.e. behaviors of P' are also behaviors of P.
- $\forall Q \ : \ ((Q \mid\sim P_\phi)$ and $(Q \sqsubseteq P)) \Longrightarrow Q \sqsubseteq P'$, i.e. all good behaviors of P are also behaviors of P'.



The rest of the paper is structured as follows. In section 2, we briefly describe the global calculus. Section 3 presents the formalism used to specify security policies. Section 4 illustrates the formalism used to specify choreographed services. In section 5, we present our security framework for monitoring a choreography. In section 6, we discuss some related works while conclusions are in section 7.

2 Global Calculus

The global calculus [10] is distilled from WS-CDL. It describes the global flow of interactions between the participants. The syntax of the global calculus is given by the standard BNF.

2.1 Syntax of the Global Calculus

$$
\begin{array}{rll}
I ::= & A \to B : ch(\nu \tilde{s}).I & (Init) \\
| & A \to B : s\langle op, e, y\rangle.I & (Comm) \\
| & x@A := e.I & (Assign) \\
| & if\ e@A\ then\ I_1\ else\ I_2 & (IfThenElse) \\
| & I_1 + I_2 & (Sum) \\
| & I_1 | I_2 & (Par) \\
| & (\nu s)I & (New) \\
| & X^A & (recVar) \\
| & rec\ X^A.I & (Rec) \\
| & 0 & (Inaction)
\end{array}
$$

The first rule $(Init)$ says that the participant A invokes a service ch located at B and initiates new freshly generated session channels $\tilde{s}$. The second rule $(Comm)$ expresses the sending action by A whose message consists of a selected operator op and an expression e to the participant B which will store the value of e at the variable y. $(Assign)$ is a local construct which updates the variable x located at A with e. (Par) and (Sum) are respectively the parallel composition and the non-deterministic choice of interactions. $(IfThenElse)$ is the standard conditional operation, while $(recVar)$ and (Rec) are used to express recursive behavior of interactions. 0 is the inactive interaction.

2.2 Semantics of the Global Calculus

The formal semantics of the global calculus is defined in terms of *configurations* (σ, I). The notation $(\sigma, I) \rightarrow (\sigma', I')$ says that the global description I at a state σ (which is the collection of all local states of the participants) will reduce to I' at a new state σ'. Samples of reduction rules are presented here while the overall operational semantic is detailed in [10].

$$(Init)\frac{-}{(\sigma, A \rightarrow B : ch(\nu \tilde{s}).I) \rightarrow (\sigma, (\nu \tilde{s})I)}$$

$$(Comm)\frac{\sigma \vdash e@A \Downarrow v}{(\sigma, A \rightarrow B : s\langle op, e, x\rangle.I) \rightarrow (\sigma[x@B \mapsto v], I)}$$

$$(Assign)\frac{\sigma \vdash e@A \Downarrow v}{(\sigma, x@A := e.I) \rightarrow (\sigma[x@A \mapsto v], I)} \quad (Par)\frac{(\sigma, I_1) \rightarrow (\sigma', I_1')}{(\sigma, I_1|I_2) \rightarrow (\sigma', I_1'|I_2)}$$

$$(Sum)\frac{(\sigma, I_1) \rightarrow (\sigma', I_1')}{(\sigma, I_1 + I_2) \rightarrow (\sigma', I_1')} \quad (Rec)\frac{(\sigma, I[rec\ X^A.I/X^A]) \rightarrow (\sigma', I')}{(\sigma, rec\ X^A.I) \rightarrow (\sigma', I')}$$

The first rule is for initiation : A invokes a service located at B and initiates session channels $\tilde{s}$ which will be shared locally by A and B. The second rule is for communication. The notation $\sigma \vdash e \Downarrow v$ is an evaluation judgment : σ evaluates the expression e to the value v. $\sigma[x@B \mapsto v]$ is the resulting state of updating the local variable x at B by v. The evaluation judgment is also used in the assignment rule. The fourth rule is for parallelism. The (Sum) rule describes the behavior of a non-deterministic choice. The (Rec) rule says that if the unfolding of $rec\ X^A.I$ under σ reduces to I' then $rec\ X^A.I$ under σ will reach (σ', I').

Example 1. This example shows a communication between a buyer and a seller. These participants share new freshly generated session channels $B2Sch$ and $S2Bch$. Through $S2Bch$, the seller offers a quote to the buyer. Through $B2Sch$, the buyer selects one of the two options offered by the seller, QuoteAccept and QuoteReject. If the first option is selected, the buyer sends the quote "100" which will be stored in x by Seller and continues with the interaction I_1. In the other case, the seller sends the abort number stored in the variable $x_{AbortNo}$ which will be stored in y by the Seller and terminates.

Buyer $\rightarrow$ Seller : $ch(\nu B2Sch, S2Bch)$.
Seller $\rightarrow$ Buyer : $S2Bch\langle quote, 100, y\rangle$.
if $y@Buyer \leq 1000$ then
  { Buyer $\rightarrow$ Seller : $B2Sch\langle quoteAccept, 100, x\rangle.I_1$ }
else
  { Buyer $\rightarrow$ Seller : $B2Sch\langle quoteReject, x_{abortNo}, x\rangle.0$ }

3 Security Policy Specification

The language that we will use for the specification of security policies is EPC^{ϕ} which is a subset of EPC (End-point calculus). The End-Point Calculus precisely identifies a local behavior of each participant in a web service. End-Point

Calculus is inspired from the π-calculus. It is an applied variant of π-calculus augmented with the notion of participants and their local states. In [10], authors have established a projection from the global calculus into the end-point calculus. So descriptions of participants behaviors in end-point calculus are not extracted directly from the choreography, but projected from the global calculus. Details of this theory of end-point projection are presented in [10]. Since in the proposed enforcement approach we transform the security policy to a monitor, we reach immediately this goal by choosing EPC^{ϕ} since it is a subset of EPC.

3.1 Syntax of EPC^{ϕ}

We show the syntax of EPC^{ϕ}, where s denote session channels, e an expression which can be an atomic value (such as a natural number or a boolean value) or a function (such as arithmetic functions and boolean operators) or a variable. $op_1, op_2, ...$ range over operations. x is a variable. The first construct is a receiving action, it is the invocation of one of the operators op_i and reception of an expression which will be evaluated and stored in x_i. The second construct is a sending action, it is the invocation of operator op with expression e. Furthermore, the operator "$\oplus$" represents the alternative composition. Finally, recursion is used for representing unbounded repetition. For representing recursive behaviors, we use term variables $X, Y, \ldots$ The operator used for recursion is rec $X.P$. Each occurrence of X in P denotes a recurring point. rec $X.P$ behaves as P until an occurrence of X is found in the execution of P, then it will return to rec $X.P$.

$$s \rhd \sum_i op_i(x_i).P_i, \quad \overline{s} \lhd op\langle e\rangle.P, \quad P_1 \oplus P_2, \quad X, \quad \text{rec } X.P, \quad 0$$

3.2 Semantics of EPC^{ϕ}

Note that processes do not evolve alone. A process is located in a participant which synchronize with another one to evolve. A participant A with its behavior P at a local state σ is called a network and denoted by $A[P]_{\sigma}$. Syntax of networks is given by the following grammar :

$$\begin{array}{rrl} N ::= & A[P]_{\sigma} & \text{(Participant)} \\ | & N|M & \text{(Parallel-NW)} \\ | & (\nu s)N & \text{(Res-NW)} \\ | & \epsilon & \text{(Inaction-NW)} \end{array}$$

where $A[P]_{\sigma}$ is a participant with its behavior as shown in the first construct. Two communicating participants are represented by two networks combined by parallel composition. When two participants initiate a session channel for communication, this session channel must be restricted to these two participants, this is given by (Res-NW). ϵ denote the lack of networks. Reduction rules of networks are given by :

$$\frac{M \equiv M' \quad M' \to N' \quad N' \equiv N}{M \to N}\text{Struct-NW}$$

$$\frac{M \to M'}{M|N \to M'|N}\text{Par-NW} \quad \frac{M \to M'}{(\nu s)M \to (\nu s)M'}\text{Res-NW}$$

Semantics of EPC^{ϕ} is then given by :

$$\frac{\sigma_2 \vdash e \Downarrow v}{A[\overline{s} \lhd op_j\langle e\rangle.P]_{\sigma_1}|B[s \rhd \sum_i op_i(x_i).Q_i]_{\sigma_2} \to A[P]_{\sigma_1}|B[Q_j]_{\sigma_2[x_j \mapsto v]}}$$

$$\frac{A[P_1]_{\sigma} \to A[P_1']_{\sigma'}}{A[P_1 \oplus P_2]_{\sigma} \to A[P_1']_{\sigma'}} \quad \frac{A[P[\text{rec } X.P/X]]_{\sigma} \to A[P']_{\sigma'}}{A[\text{rec } X.P]_{\sigma} \to A[P']_{\sigma'}}$$

Security policies are usually specified in logic-based languages. Such languages employ mainly three operators to compose properties : *and, or* and *not*. The *or* operator is given here by $\oplus$. The *not* operator is defined here as follows : assume we have two participants A and B communicating and we want to apply the property "A should not send op_1 to participant B", then we write it using EPC^{ϕ} as follows :

$$P = \text{rec } X.(\overline{s_B} \lhd \bigoplus_{i \neq 1} op_i\langle e_i\rangle.X \oplus s_B \rhd \sum_i op_i(x_i).X)$$

Example 2. Assume we have a client wanting to check its account details within a bank. The bank should not send him back his account details if he is not yet authenticated. A client is said to be authenticated if he has received from the bank an acceptance for his authentication's attempt using the operation *accept*. The bank answers the client for his account request through the operation $resAccount$. The security property will then be written as follows :

$$P_{\phi} = \text{rec } X.(s \rhd \sum_i op_i(x_i).X \quad \oplus$$

$$\overline{s} \lhd \bigoplus_{op_i \notin \{accept, resAccount\}} op_i\langle e\rangle.X \quad \oplus$$

$$\overline{s} \lhd accept.\text{rec } Y.(\overline{s} \lhd \bigoplus_i op_i\langle e_i\rangle.Y \oplus s \rhd \sum_i op_i(x_i).Y))$$

The security property is expressed using the recursion operator. In the recursion block, we have 3 behaviors combined with the internal choice $\oplus$ which we denote by P_{ϕ_1}, P_{ϕ_2} and P_{ϕ_3} where

$$P_{\phi_1} = s \rhd \sum_i op_i(x_i).X$$

$$P_{\phi_2} = \overline{s} \lhd \bigoplus_{op_i \notin \{accept, resAccount\}} op_i\langle e\rangle.X$$

$$P_{\phi_3} = \overline{s} \lhd accept.\text{rec } Y.(\overline{s} \lhd \bigoplus_i op_i\langle e_i\rangle.Y \oplus s \rhd \sum_i op_i(x_i).Y)$$

The block P_{ϕ_1} expresses the fact that the property allows all receiving actions. The block P_{ϕ_2} inhibits sending *accept* or $resAccount$ to the client. The block P_{ϕ_3} intercepts the *accept* sending action and then no more restrictions are imposed since the authentication of the client is accepted.
So, an intruder who wants to check an account's details without authentication cannot achieve its goal since the $resAccount$ is only permitted in the block following the *accept* sending action.

4 Choreography Specification

The technique used in this paper for describing the composition of web services is the choreography. The WS-CDL is based on two engineering principles :

Service Channel Principle corresponds to the repeated availability of service channels.

Session Principle is a basic principle in many communication-centred programs which says a sequence of conversations belonging to a protocol should not be confused with other concurrent runs of this or other protocols by the participants.

As a specification of the choreography description language WS-CDL, we introduce a modified version of the end-point calculus which is a formalism presented in [10]. The end-point calculus describes the behavior of a participant in a choreography from its end-point view.

4.1 Secured End-Point Calculus

The secured end-point calculus EPC_S^ϕ is a variant of EPC (End-Point Calculus)[10]. EPC_S^ϕ has the particularity of explicitly handling the monitoring concept through its operator ∂_{P_ϕ}. For instance, the process $\partial_{P_\phi}(P)$ can execute only actions that could be executed by both the controller P_ϕ and the process P. We describe hereafter the formal syntax and dynamic semantics of the secured end-point calculus.

Syntax of EPC_S^ϕ Since EPC^ϕ is a subset of EPC and EPC_S^ϕ is an extension of EPC, a part of the syntax of EPC_S^ϕ is presented in the last section. So in this section we will present the overall syntax of EPC_S^ϕ but we will describe only constructs that have not been described before.

$$
\begin{array}{rl}
P ::= & !ch(\tilde{s}).P \\
| & \overline{ch}(\nu\tilde{s}).Q \\
| & s \rhd \sum_i op_i(x_i).P_i \\
| & \overline{s} \lhd op\langle e\rangle.Q \\
| & x := e.P \\
| & \text{if } e \text{ then } P \text{ else } Q \\
| & P \oplus Q \\
| & P \mid Q \\
| & (\nu s)P \\
| & X \\
| & \text{rec } X.P \\
| & 0 \\
| & \hline \\
| & \partial_{P_\phi}(P)
\end{array}
$$

The two first constructs represent session initiation. $!ch(\tilde{s}).P$ is used for input and $\overline{ch}(\nu\tilde{s}).Q$ for output. $!ch(\tilde{s})$ says that the service channel ch, which is available to public, is ready to receive an unbounded number of invocations, offering a communication via its freshly generated session channels $s \in \tilde{s}$. $\overline{ch}(\nu\tilde{s})$ is an invocation of a service located at the service channel ch and an initiation of a communication session which will occur through session channels $\tilde{s}$. After a session has been initiated between two participants and freshly generated session channels have been shared between them, they can communicate using the communication constructs. The assignment operator is $x := e.P$. if e then P else Q is a choice based on the evaluation of the expression e. The parallel composition is given by $P|Q$. The restriction construct $(\nu s)P$ indicates that the session channel s is local to P. 0 denotes the lack of actions. Finally, $\partial_{P_\phi}(P)$ is the enforcement operator which controls the execution of P by allowing it to evolve if P_ϕ is satisfied.

Semantics of EPC_S^ϕ

Initiation/Communication simulation Let P be a process stipulating the local behavior of a participant A in a web service. We observe in the end-point calculus reduction rules that processes always evolve without an external factor unless in initiation and communication. In fact, a participant initiating a communication or communicating with an another participant needs to synchronize with him to realize the interaction . Therefore, for enforcing security policies on a participant's behavior, we need to have a totally local description of its interactions. So we need to define a simulation relation for the initiation and the communication of processes so that we can simulate the evolution of a process locally, i.e. without synchronizing with an another participant . We will define the normal form of a process then introduce the simulation relation.

Definition 1 (Normal form of a process). *Every process representing the local behavior of a participant in a web service can be written as an internal sum of processes, which we call the normal form of a process :*

$$\forall P \in \mathcal{P},\ P = \bigoplus_i a_i P_i$$

where $\mathcal{P}$ denotes the set of processes, a_i range over atomic actions and P_i range over processes in $\mathcal{P}$.

Definition 2 (Simulation Relation). *We define a simulation relation over networks, denoted by $A[P]_\sigma \overset{a}{\rightsquigarrow} A[P']_{\sigma'}$, which says that process P in A at the state σ is able to execute the action a and reduce to P' with a new local state σ'. The simulation relation is defined following this rule*

$$\frac{P = \bigoplus_i a_i P_i \quad \exists i \in \{1, \ldots, n\} : a = a_i}{A[P]_\sigma \overset{a}{\rightsquigarrow} A[P_i]_\sigma}$$

This rule says that when P is written in its normal form and one of the constituting processes is able to do an action a then $A[P]_\sigma$ is also able to do it.

Semantics of EPC_S^{ϕ} :

$$\text{Init}\ \frac{}{A[!ch(\tilde{s}).P \mid P']_{\sigma_A} \mid B[\overline{ch}(\nu\tilde{s}).Q \mid Q']_{\sigma_B} \to (\nu\tilde{s})(A[!ch(\tilde{s}).P|P|P']_{\sigma_A} \mid B[Q|Q']_{\sigma_B})}$$

$$\frac{\sigma_A \vdash e \Downarrow v}{A[x := e.P|P']_{\sigma_A} \to A[P|P']_{\sigma_A[x\mapsto v]}}\text{Assign}$$

$$\frac{\sigma_A \vdash e \Downarrow tt}{A[\text{if } e \text{ then } P_1 \text{ else } P_2|P']_{\sigma_A} \to A[P_1|P']_{\sigma_A}}\text{IfTrue}$$

$$\frac{\sigma_A \vdash e \Downarrow ff}{A[\text{if } e \text{ then } P_1 \text{ else } P_2|P']_{\sigma_A} \to A[P_2|P']_{\sigma_A}}\text{IfFalse}$$

$$\frac{A[P_1|R]_{\sigma_A} \to A[P_1'|R]_{\sigma_A'}}{A[P_1|P_2|R]_{\sigma_A} \to A[P_1'|P_2|R]_{\sigma_A'}}\text{Par} \qquad \frac{A[P]_{\sigma_A} \to A[P']_{\sigma_A'}}{A[(\nu s)P]_{\sigma_A} \to A[(\nu s)P']_{\sigma_A'}}\text{Res}$$

$$\text{Init-In-Sec}\ \frac{}{A[\partial_{P_\phi}(!ch(\tilde{s}).P)]_{\sigma_A}|B[\overline{ch}(\nu\tilde{s}).Q|R]_{\sigma_B} \to (\nu\tilde{s})(A[\partial_{P_\phi}(P)|\partial_{P_\phi}(!ch(\tilde{s}).P)]_{\sigma_A}|B[Q|R])}$$

$$\text{Init-Out-Sec}\ \frac{}{A[\partial_{P_\phi}(\overline{ch}(\nu\tilde{s}).P)]_{\sigma_A}|B[!ch(\tilde{s}).Q|R]_{\sigma_B} \to (\nu\tilde{s})(A[\partial_{P_\phi}(P)]_{\sigma_A}|B[!ch(\tilde{s}).Q|Q|R]_{\sigma_B})}$$

$$\frac{A[P]_{\sigma_A} \overset{s\triangleright op(x)}{\rightsquigarrow} A[P']_{\sigma_A} \quad A[P_\phi]_{\sigma_A} \overset{s\triangleright op(x)}{\rightsquigarrow} A[P'_\phi]_{\sigma_A} \quad \sigma_A \vdash e \Downarrow v}{A[\partial_{P_\phi}(P)]_{\sigma_A}|B[\overline{s} \lhd op\langle e\rangle.Q|R]_{\sigma_B} \to A[\partial_{P'_\phi}(P')]_{\sigma_A[x\mapsto v]}|B[Q|R]_{\sigma_B}}\text{Comm-In-Sec}$$

$$\text{Comm-Out-Sec}$$

$$\frac{A[P]_{\sigma_A} \overset{\overline{s}\lhd op_j\langle e\rangle}{\rightsquigarrow} A[P']_{\sigma_A} \quad A[P_\phi]_{\sigma_A} \overset{\overline{s}\lhd op_j\langle e\rangle}{\rightsquigarrow} A[P'_\phi]_{\sigma_A} \quad \sigma_B \vdash e \Downarrow v}{A[\partial_{P_\phi}(P)]_{\sigma_A}|B[s \rhd \sum_i op_i(x_i).Q_i|R]_{\sigma_B} \to A[\partial_{P'_\phi}(P')]_{\sigma_A}|B[Q_j|R]_{\sigma_B[x_j\mapsto v]}}$$

$$\frac{\sigma_A \vdash e \Downarrow v}{A[\partial_{P_\phi}(x := e.P)]_{\sigma_A} \to A[\partial_{P_\phi}(P)]_{\sigma_A[x\mapsto v]}}\text{Assign-Sec}$$

$$\frac{A[\partial_{P_\phi}(P_1)]_\sigma \to A[\partial_{P'_\phi}(P'_1)]_{\sigma'}}{A[\partial_{P_\phi}(P_1|P_2)]_\sigma \to A[\partial_{P'_\phi}(P'_1|P_2)]_{\sigma'}}\text{Par-Sec}$$

$$\frac{\sigma \vdash e \Downarrow tt}{A[\partial_{P_\phi}(P|if\ e\ then\ P_1\ else\ P_2)]_\sigma \to A[\partial_{P_\phi}(P|P_1)]_\sigma}\text{IfTrue-Sec}$$

$$\frac{\sigma \vdash e \Downarrow ff}{A[\partial_{P_\phi}(P|if\ e\ then\ P_1\ else\ P_2)]_\sigma \to A[\partial_{P_\phi}(P|P_2)]_\sigma}\text{IfFalse-Sec}$$

$$\frac{A[\partial_{P_\phi}(P)]_\sigma \to A[\partial_{P'_\phi}(P')]_{\sigma'}}{A[(\nu\tilde{s})(\partial_{P_\phi}(P))]_\sigma \to A[(\nu\tilde{s})(\partial_{P'_\phi}(P')]_{\sigma'})}\text{Res-Sec}$$

The Init-rule shows how two participants initiate a session by sharing new freshly generated session channels $\tilde{s}$. These session channels are restricted to participants A and B by the binding operator (ν). Assignment is a local construct. Assign-rule evaluates an expression e and assigns the result of this evaluation to the variable x in A, then A behaves as P. The Res-rule restricts the use of session channels $\tilde{s}$ to the process P in A. Init-In-Sec and Init-Out-Sec are the rules for communication initiation. We do not control session initiations but we control communication messages between the participants. Communication rules say if P_ϕ and P are able to send or receive through the same session channel the same operation and become respectively P'_ϕ and P' then $\partial_{P_\phi}(P)$ do this action and becomes $\partial_{P'_\phi}(P')$. Secured assignment rule says that assignment is not considered by enforcement. The secured parallel composition rule says that the security operator applied to $P_1|P_2$ can evolve into $(\partial_{P'_\phi}(P'_1|P_2))$ if P_1 can evolve simultaneously with the policy security P_ϕ into respectively P'_1 and P'_ϕ. For the restriction rule, binding session channels does not affect the enforcement. Finally, the conditional rules say that enforcement is not affected by conditionals.

5 Choreography Monitoring

The goal of this research is to enforce security policies over a choreography. Some of the important features of the enforcement operator is that it allows us to enforce only concerned participants by the security policies. In this in-lined monitoring framework, we do not modify the original behaviors of participants in a choreography. But if the security policy is not verified the evolution of the choreography will stop. In this section, we prove the correctness of our theory by defining first some notions such as the partial order over processes and satisfaction notions.

Definition 3 (Partial Order over Processes). *Let $A[P_1]_\sigma$, $A[P_2]_\sigma$ be two networks. We say $A[P_1]_\sigma \sqsubseteq A[P_2]_\sigma$ if the following condition hold :*

$$A[P_1]_\sigma \overset{a}{\rightsquigarrow} A[P'_1]_\sigma \implies A[P_2]_\sigma \overset{a}{\rightsquigarrow} A[P'_2]_\sigma \text{ and } A[P'_1]_\sigma \sqsubseteq A[P'_2]_\sigma.$$

Definition 4 (Satisfaction Notions). *Let P_ϕ be a security policy and ξ a trace. Symbols $\vDash$ and $|\sim$ are defined as follows :*

- *We say that ξ satisfies P_ϕ, denoted by $\xi \vDash P_\phi$, if $\xi \in [\![P_\phi]\!]$ where $[\![P_\phi]\!]$ denotes the set of traces of P_ϕ.*
- *We say that ξ could satisfy P_ϕ, denoted by $\xi \mid\sim P_\phi$, if it exists a trace ξ' such that $\xi.\xi' \vDash P_\phi$.*

Theorem 1. *Let P be a process and P_ϕ a policy security. The following properties hold :*

1. $\partial_{P_\phi}(P) \mid\sim P_\phi$,
2. $A[\partial_{P_\phi}(P)]_\sigma \sqsubseteq A[P]_\sigma$,
3. $\forall P' \ : \ ((P' \mid\sim P_\phi)$ *and* $(A[P']_\sigma \sqsubseteq A[P]_\sigma)) \Longrightarrow A[P']_\sigma \sqsubseteq A[\partial_{P_\phi}(P)]_\sigma$.

Proof. 1. The process $\partial_{P_\phi}(P)$ is defined so that it can evolve into another process only if the security policy P_ϕ is satisfied.
2. As well as the first property, the proof is obtained directly from the reduction rules of the security operator and the definition of the partial order.
3. Consider a process $P' \in \mathcal{P}$ such that $P' \mid\sim P_\phi$ and $A[P']_\sigma \sqsubseteq A[P]_\sigma$. Suppose $A[P']_\sigma \overset{a}{\rightsquigarrow} A[P'_1]_{\sigma'}$. Since $A[P']_\sigma \sqsubseteq A[P]_{\sigma'}$, we conclude directly from the definition of $\sqsubseteq$ that $A[P]_\sigma \overset{a}{\rightsquigarrow} A[P_1]_{\sigma'}$. Since $P' \mid\sim P_\phi$, we can also conclude from the definition of $\mid\sim$, that $a \mid\sim P_\phi$. Finally, as $A[P]_\sigma \overset{a}{\rightsquigarrow} A[P_1]_{\sigma'}$ and $a \mid\sim P_\phi$ so $A[\partial_{P_\phi}(P)]_\sigma \overset{a}{\rightsquigarrow} A[\partial_{P'_\phi}(P_1)]_{\sigma'}$ and then $A[P']_\sigma \sqsubseteq A[\partial_{P_\phi}(P)]_\sigma$.

Example 3 (Buyer-Seller Protocol). The buyer initiates a communication with the seller and requests for a quote. The seller sends back the quote. If the buyer rejects it then the protocol terminates. Otherwise, the seller sends to the buyer an order confirmation and the buyer confirms his command. In this case, the seller contacts the shipper and asks for delivery details which he transfer to the buyer and the protocol terminates. The security property that we want to apply in this protocol is : "the seller should communicate with the shipper only if the buyer confirms his command". As a consequence, the seller won't send to buyer the delivery details if he has not confirmed his command. The protocol is depicted in Fig.1. Critical actions that we have to supervise are in dark grey in Fig.1.

The choreography's description in global calculus is the following :

Buyer $\rightarrow$ Seller : $B2SCh(s)$.
Seller $\rightarrow$ Buyer : $s\langle ackSession\rangle$.
Buyer $\rightarrow$ Seller : $s\langle reqQuote\rangle$.
Seller $\rightarrow$ Buyer : $s\langle respQuote, v_{quote}, x_{quote}\rangle$.
if $resonable(x_{quote})$@Buyer then
  Buyer $\rightarrow$ Seller : $s\langle quoteAccept\rangle$.
  Seller $\rightarrow$ Buyer : $s\langle orderConfirm\rangle$.
  Buyer $\rightarrow$ Seller : $s\langle ackConfirm\rangle$.
  Seller $\rightarrow$ Shipper : $B2SHCh(s')$.
  Shipper $\rightarrow$ Seller : $s'\langle ackSession\rangle$.
  Seller $\rightarrow$ Shipper : $s'\langle reqDelivDet\rangle$.
  Shipper $\rightarrow$ Seller : $s'\langle delivDet, v_{delivDet}, x_{delivDet}\rangle$.
  Seller $\rightarrow$ Buyer : $s\langle delivDet, x_{delivDet}, x_{delivDet}\rangle.0$
else
  Buyer $\rightarrow$ Seller : $s\langle quoteReject\rangle.0$

The end-point projection gives the end-point behaviors. The buyer's behavior is the following :

$$Buyer[\overline{B2SCh}(\nu s).s \triangleright ackSession.\overline{s} \triangleleft reqQuote.s \triangleright respQuote(x_{quote})$$

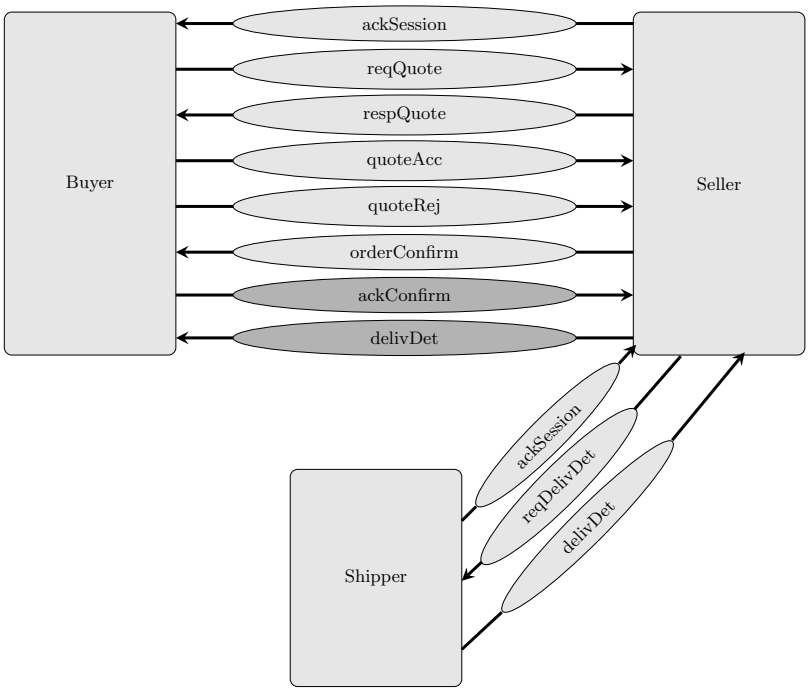


Fig. 1. Buyer Seller Protocol

.if $reasonable(v_{quote})$ then $\overline{s} \lhd quoteAcc.s \rhd orderConfirm.\overline{s} \lhd ackConfirm$

$.s \rhd delivDet(x_{delivDet}).0$ else $\overline{s} \lhd quoteRej.0]$

The seller's behavior is the following :

$$Seller[B2SCh(s).\overline{s} \lhd ackSession.s \rhd reqQuote.\overline{s} \lhd respQuote\langle e_{quote}\rangle.$$

$$(s \rhd quoteAcc.\overline{s} \lhd orderConfirm.s \rhd ackConfirm.$$

$$\overline{B2SHCh}(\nu s').s' \rhd ackSession.\overline{s'} \lhd reqDelivDet\langle e_{buyer}\rangle.s' \rhd delivDet(x_{delivDet})$$

$$.\overline{s} \lhd delivDet(x_{delivDet}).0\ ) + (s \rhd quoteRej.0)]$$

The shipper's behavior is the following :

$$Shipper[B2SHCh(s').\overline{s'} \lhd ackSession.s' \rhd reqDelivDet(x_{buyer})$$

$$.\overline{s'} \lhd delivDet\langle e_{delivDet}\rangle.0]$$

The security property has the following behavior :

$$
\begin{aligned}
P_\phi = \text{rec } X.(\ & \overline{s} \lhd \bigoplus_{op \neq delivDet} op_i\langle e\rangle.X \quad \oplus \\
& s \rhd \sum_{op \neq ackConfirm} op_i(x_i).X \quad \oplus \\
& s \rhd ackConfirm.\text{rec } Y.(\overline{s'} \lhd \bigoplus_i op_i\langle e\rangle.Y \oplus s' \rhd \sum_i op_i(x_i).Y \\
& \oplus \overline{s} \lhd \bigoplus_i op_i\langle e\rangle.Y) \\
)\ &
\end{aligned}
$$

The security property P_ϕ is written using the recursion operator. In the recursion block, there are 3 processes combined with the internal choice. The first process is : $\overline{s} \lhd \bigoplus_{op \neq delivDet} op_i\langle e\rangle.X$ which says that, through the session channel s (shared between the buyer and the seller), the buyer can execute any sending action unless delivery details which should be done after the seller has received a confirmation from the buyer. The second process is : $s \rhd \sum_{op \neq ackConfirm} op_i(x_i).X$ which says that the seller can receive any operation other than $ackConfirm$. This action is intercepted in the third process $s \rhd ackConfirm.\text{rec } Y.(\overline{s'} \lhd \bigoplus_i op_i\langle e\rangle.Y \oplus s' \rhd \sum_i op_i(x_i).Y \oplus \overline{s} \lhd \bigoplus_i op_i\langle e\rangle.Y)$. In this process, after the seller has received a confirmation from the buyer, he can communicate with the shipper through the session channel s' without any restriction and he has also no restriction on his sending actions to the buyer, so he can send him the delivery details.

6 Related Works

Web services verification have been a subject of interest of several research efforts. Some of the relevant contributions in this domain are cited in this section. Most of formal approaches introduced a monitor which does not stop the program when a violation is detected. Moreover, these contributions implement a monitor as a web service in addition to other web services. The originality of our work is the introduction of the monitor within concerned participants processes. In [11], a run-time event-based approach to deal with the problem of monitoring conformance of interaction sequences is presented. When a violation is detected, the program shows errors in dashboards. In [12], authors introduce an approach to verify the conformance of a web service implementation against a behavioral specification, through the application of testing. The Stream X-machines are used as an intuitive modeling formalism for constructing the behavioral specification of a stateful web service and a method for deriving test cases from that specification in an automated way. The test generation method produces complete sets of test cases that, under certain assumptions, are guaranteed to reveal all non-conformance faults in a service implementation under test. However, this approach only returns non-conformance faults and does not react dynamically against these errors. While, in [13], authors propose a monitoring framework of a choreographed service which supports the early detection of faults and decide

whether it is still possible to continue the service. Authors in [**?**] have proposed service automata as a framework for enforcing security policies in distributed systems. They encapsulate the program in a service automaton composed of the monitored program, an interceptor, an enforcer, a coordinator and a local policy. The interceptor intercepts critical actions and passes them to the coordinator that determines whether the action complies the security policy or not and decides upon possible countermeasures then the enforcer implements these decisions. However the authors do not precise how to detect critical actions.

7 Conclusion

In this paper, we have introduced a formal approach allowing to automatically enforce a security policy on choreographed services. Indeed, we introduced a new calculus with an enforcement operator ∂_{P_ϕ}. The semantics of the proposed calculus insure that choreographed services can evolve only if it does not violate the enforced security policy. The originality of our work consists on the fact that we do not add a new web service as a monitor but rather we wrap the security policy inside the choreographed services.

Future work will focus on the definition of a complete mapping between WS-CDL and global calculus. Moreover, we will seek means to optimize the enforced choreographed services so that we reduce as much as we can the overhead due to the enforcement operator.

References

1. Corporation, I.: Business process execution language for web services bpel-4ws. http://www.ibm.com/developerworks/library/ws-bpel/ (2002)
2. F.Leymann: Web services flow language (wsfl) version 1.0. Technical Report, International Business Machines Corporation (IBM) (May 2001)
3. Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y.: Web services choreography description language version 1.0. W3C Working Draft (December 2004)
4. Langar, M., Mejri, M., Adi, K.: Formal enforcement of security policies on concurrent systems. J. Symb. Comput. **46**(9) (2011) 997–1016
5. Khoury, R., Tawbi, N.: Corrective enforcement: A new paradigm of security policy enforcement by monitors. ACM Trans. Inf. Syst. Secur. **15**(2) (2012) 10
6. Yi, X., Kochut, K.: A cp-nets-based design and verification framework for web services composition. In: ICWS. (2004) 756–760
7. Hamadi, R., Benatallah, B.: A petri net-based model for web service composition. In: ADC. (2003) 191–200
8. Salaün, G., Bordeaux, L., Schaerf, M.: Describing and reasoning on web services using process algebra. In: ICWS. (2004) 43–
9. Kazhamiakin, R., Pandya, P.K., Pistore, M.: Timed modelling and analysis in web service compositions. In: ARES. (2006) 840–846
10. Carbone, M., Honda, K., Yoshida, N.: Structured communication-centred programming for web services. In: ESOP. (2007) 2–17

11. Baouab, A., Perrin, O., Godart, C.: An optimized derivation of event queries to monitor choreography violations. In: ICSOC. (2012) 222–236
12. Dranidis, D., Ramollari, E., Kourtesis, D.: Run-time verification of behavioural conformance for conversational web services. In: ECOWS. (2009) 139–147
13. Ardissono, L., Furnari, R., Goy, A., Petrone, G., Segnan, M.: Monitoring choreographed services. In: Innovations and Advanced Techniques in Computer and Information Sciences and Engineering. (2007) 283–288
14. Fu, X., Bultan, T., Su, J.: Analysis of interacting bpel web services. In: WWW. (2004) 621–630

A Timestamping Scheme with Eternal Security in the Bounded Storage Model

Assia Ben Shil[1]
Faculty of Sciences of Bizerte
University of Carthage
Email: essia.benshil@gmail.com

and

Kaouther Blibech Sinaoui[2]
ISSAT of Mateur
University of Carthage
Email: kaouther.blibech@gmail.com

[1,2] LIP2 Laboratory, Tunis, Tunisia

Abstract. Digital timestamping is a cryptographic technique allowing affixing a reliable date to a digital document. The security of most existing timestamping systems is based on the security of the used cryptographic techniques as hash functions. However, a hash function has a limited lifetime. In this context, we provide a non-interactive timestamping scheme in the bounded storage model (BSM) whose security is not related to the life of any cryptographic technique. We prove, in fact, that our timestamping scheme is eternally secure even against an adversary with unlimited computing power.

Keywords: Timestamping, bounded storage model, computing power, eternal security.

1 Introduction

In the last years, digital documents are beginning to replace paper documents in several areas. For this reason, it is important to locate a digital document in time to prove its existence and integrity since a given date. This task is achieved through a timestamping system that operates in two phases: a timestamping phase allowing one or more Timestamping Authority (TSA) to affix a reliable date to a document and generate the associated timestamp and a verification phase allowing any potential verifier to verify the correctness of the produced timestamp.

The security of most existing timestamping systems [2-10] is based on the security of the used cryptographic techniques as hash functions. These techniques are secure under the assumption saying that the users' computing power

is limited. However, nowadays, the computing power of computers grows exponentially. Therefore, the used hash functions are not secure all the time [17]. So, the existing timestamping systems cannot be considered secure forever. To propose timestamping systems producing timestamps with eternal validity, we are placed in the Bounded Storage Model (BSM) [13, 14]. In this model, Maurer assumes that users' storage capacity is limited, while their computing power can be unlimited. In this model, the ciphers are eternally secure [1]. The principle is that a very long random string is transmitted in every round t (the interval between the time t and the time $t+1$). At the time of the transmission of this string, no participant has sufficient storage capacity to store it fully.Even if later, his storage capacity becomes sufficient to fully store the string transmitted in t, the user has already lost he access to this string. Thus, the performed encryptions are valid forever. Moran proposed in [15] a non-interactive timestamping scheme in the bounded storage model. Indeed, in the scheme of Moran, each stamper or document owner can timestamp his document locally without communicating with any third party [15], unlike conventional timestamping systems that involve at least one TSA. Thus, the non-interactive timestamping ensures total confidentiality and hides even the fact that a timestamping occurred. These characteristics have made the proposition of Moran very interesting, but the timestamping system he proposed suffers from a lack of precision and practical details. In this context, we present a non-interactive timestamping scheme in the BSM. This paper is organized as follows: After introducing the BSM in section 2, we present firstly Moran's timestamping system, and then our non-interactive timestamping scheme in the BSM in section 3. Then, we detail more formally our timestamping scheme. Finally, we formally prove the eternal security of our timestamping solution.

2 The Bounded Storage Model

Generally, in cryptography, the proposed systems and the used functions are secure under the assumption that there is a limit of the computational power of any user or adversary. In the bounded storage model, the proposed systems must be secure even against an adversary with an unlimited computational power. The BSM was proposed by Maurer in 1992 [13] for the development of encryption keys. It aims to generate, from a short secret key K, a key with a large size X [14] that can be used as encryption key. The system operates as follows: In this model, it is assumed that the storage capacity is unlimited, no assumptions about the computing power was made. Let s be the assumed limit on a user's storage capacity. Ciphers in the BSM use a very long string R called randomizer. The latter may for example be a random sequence of bits transmitted by a satellite. If R is a random string of r bits, the space of R is $\{0,1\}^r$. Notice that $r \gg s$ is required to ensure that no user can fully store R. Having a secret key K of size k in the space $\{0,1\}^k$, we can use a known function $f : \{0,1\}^r \times \{0,1\}^k, x \rightarrow \{0,1\}$ to generate the derived key $X = f(R,K)$ of size x bits. The function f must use only a small part of R so that we do not need to fully read R. Maurer's system

has been the subject of intensive studies. Indeed, many key generation systems have been proposed in the BSM [11]. The BSM was used for timestamping by Moran in [15]. In [3], we proposed an improvement of Moran's timestamping system. In the following section, we present the timestamping system of Moran and then our proposition is presented.

3 Timestamping Solutions in the BSM

In the BSM, we assume that a long random string R of r bits is transmitted during the round t (between t and $t+1$). If s is the maximum storage capacity of any entity in t, then $s \ll r$. Notice the space of R is $\{0,1\}^r$ where r is the size of the string R. Similarly, we consider that a document D of size d has a value in $\{0,1\}^d$. In the timestamping scheme proposed by Moran, to timestamp a document D, its content is used to select a few blocks from R whose values will be inserted in the timestamp T. Any verifier must save randomly some blocks of R (using a function named $Sketch(R)$) in order to verify, later, the validity of any timestamp made during the round t. Verifying the validity of a timestamp associated to a document D is performed at a later date by a verifier who has simply to verify that there are no conflicts between his sketch and the timestamp of D. However, in this solution, a timestamp includes some additional values of R. If the verifier cannot store during the round t more than s bits of R, he may at the verification time store s' with $s' > s$. Each verification of a timestamp generated in the round t can lead him to discover new blocks of R. After a number of verification processes, he can reconstruct partially, if not entirely R and backdate any document.

To remedy this problem, we propose a timestamping protocol allowing verifying the timestamp of a document D without learning additional values of R. To this aim, instead of inserting the blocks of R in the timestamp, these blocks will be the secret of the stamper. The verifier must then prove that he has the value of a given block in his sketch and the stamper has to prove that he has used this value to create the timestamp. In the verification process of this scheme, a verifier may accept or reject a timestamp without discovering any additional information about the string R transmitted during the round t. The idea is to timestamp the document D locally using the secret sharing scheme of Shamir [18] in order to divide D into n shares D_i. Assuming that the blocks of R are indexed by a number beginning from 1 till the number of blocks, the values R_{D_i}, indexed by the shares D_i for $1 \leq i \leq n$ are recovered. The polynomial that passes through the points $P_i(R_{D_i}, D_i)$ for $1 \leq i \leq n$ represents the timestamp of D. Note $Share(D)$ the set of shares D_i. Any user of the system saves a random subset of R named $Sketch(R)$ formed by a number of couples of values (index of block, value of the block). To verify the validity of the timestamp T of D for a random string R the verifier who stored $Sketch(R)$ proceeds as follows: For each index in both $Share(D)$ and $Sketch(R)$ he recovers the associated block R_i of R, computes its associated index by the polynomial given in the timestamp and verifies that the associated value in $Sketch(R)$ is R_i. In the following sections,

we provide a formal representation of our timestamping system and we prove its eternal security. To this aim, we will show that once the string R is transmitted, the probability of forging a fake timestamp for a document D is close to zero.

4 Definitions and Notations

In this section, we introduce some notations that we will use later in this paper.

- Randomizer: We call "randomizer" and we denote R the random string transmitted during a round t. This string is divided into n blocks of size b bits indexed from 1 to n and denoted $R_1, \ldots, R_n$. Knowing that the length of R is r bits and the size of a block is b bits, $r = n.b$. For each subset $S \subseteq I(R)$ where $I(R)$ is the set of indexes of blocks R_i $(1 \leq i \leq n)$, we denote $R_{|S}$ the set of blocks of $R \; \forall$ i $\in S$.

- Hamming Distance: A vector being a set of blocks, we define the Hamming Distance between two vectors c_1 and c_2 denoted DH as the number of blocks for which the two vectors differ.

- Threshold secret sharing: The threshold secret sharing is a technique for dividing a secret S into l shares such that the coalition of at least λ shares is necessary to reconstruct the secret while the coalition of at most $\lambda - 1$ shares do not reveal even partially the secret $(\lambda < l)$. A λ-out of-l threshold secret sharing scheme is denoted $SSS(\lambda, l)$.

- Shamir secret sharing scheme: The secret sharing scheme based on Shamir's polynomial interpolation scheme is a $SSS(\lambda, l)$. We denote it SSSS (λ, l). The principle is to fix a polynomial P of degree $\lambda - 1$ and X a set of values $(X = X_1, \ldots, X_l)$. The secret sharing function denoted Share takes as input the secret S and generates the set of shares $Share(S) = (S_1, \ldots, S_l)$ such that $\forall$i, $1 \leq i \leq l$, $S_i = P(X_i)$. The Reconstruction function denoted $Share^{-1}$ take as input a subset of X denoted $X_S = [X_{S_1}, \ldots, X_{S_\lambda}]$ such that $|X_S| = \lambda$ and the λ associated shares: $Share^{-1}(X_S, S_{X_{S_1}}, \ldots, S_{X_{S_\lambda}}) = P$, with P a polynomial such that $\forall$ $X_i \in X_S$ and $S_i \in [S_{X_{S_1}}, \ldots, S_{X_{S_\lambda}}]$, $P(X_i) = S_i$. The secret S is computed as follows: $S = P(0)$.

5 Presentation of our Timestamping Scheme

5.1 Timestamping Phase

A stamper is represented by the two following functions:

- $Store(D, R)$ that uses Shamir's secret sharing process to compute $Share(D)$ for a given document D. Then, It computes the vector $R_{|Share(D)}$ and stores it. More formally, $Store(D, R)$ consists in computing $Share(D) = (D_1, \ldots, D_l)$, where D_i is the $i^{th} index specified by D. To the vector(R_{D_1}, \ldots, R_{D_l})$ is associated the vector $Share(D)$ where R_{D_i} is the block of R indexed by D_i. We denote this vector $R_{|Share(D)}$, where the notation $R_{|I}$ means the values of blocks of R indexed by $I_1, \ldots, I_n$, with $I = I_1, \ldots, I_n$.

Definition 1. $Store(D, R) = R_{|Share(D)}$.

- $Stamp(D, Store(D, R))$ uses Shamir's reconstruction process to find the unique polynomial passing through the points $P_i(x, y)$ where x is a block of the vector $R_{|Share(D)}$ and y the associated blocks in $Share(D)$. This polynomial is the timestamp T.

Definition 2. $T = Share^{-1}(R_{|Share(D)}, Share(D))$.

5.2 Verification Phase

A verifier is represented by the two following functions:

- $Sketch(R)$ allows choosing, randomly, a set of indexes denoted H with $H \subset I(R)$, computing $R_{|H}$ and storing this vector.

Definition 3. $Sketch(R) = (H, R_{|H})$.

- $Verify(Sketch(R), D, T)$ allows verifying that there are no conflicts between $Sketch(R)$ and T.

Definition 4. *$Verify(Sketch(R), D, T)$ allows verifying the following equality : $T(R_{|H \cap Share(D)}) = H \cap Share(D)$*

In other words:

Definition 5. *$Verify(Sketch(R), D, T)$ allows verifying that $DH(T(R_{|H} \cap Share(D)), H \cap Share(D)) = 0$ for a timestamp T. If this equality is verified, the timestamp T is accepted by the verifier and is said "valid".*

5.3 The behavior of an adversary

An adversary consists in the two following functions:

- $Store^*(R)$ which saves a subset of R called C. The difference between $Sketch(R)$ and $Store^*(R)$ is that $Sketch(R)$ is computed "online" while $Store^*(R)$ function may not be.

- $Stamp^*(D, C)$ that given a document D and a string C tries to produce a timestamp T^* of D.

Definition 6. $Stamp^*(D, C) = {R^*}_{|Share(D)}$.

Definition 7. $T^* = Share^{-1}({R^*}_{|Share(D)}, Share(D))$.

Where ${R^*}_{|Share(D)}$ is the vector of blocks associated to $Share(D)$ according to T^*. If an adversary A produces for a document D and a randomizer R, a timestamp T^* that is equal to the timestamp T produced by $Stamp(D, Store(D, R))$, we say that he backdates "correctly" the document. More formally:

Definition 8. *An adversary backdates a document D with a success probability γ for a given randomizer R if : $Pr[Verify(Sketch(R), D, Stamp^*(D, Store^*(R)))] \geq \gamma$*

Definition 9. *An adversary backdates correctly a document D for a given randomizer R if $DH(V_1, V_2) = 0$ with $V_1 = R^*_{|Share(D)}$ and $V_2 = R_{|Share(D)}$.*

Definition 10. *An adversary backdates correctly a document D for a given randomizer R with at most err errors, if $DH(V_1, V_2) \leq err$, with $V_1 = R^*_{|Share(D)}$ and $V_2 = R_{|Share(D)}$.*

6 Security Proofs of our Timestamping Scheme

Given the following parameters:

- s : the storage capacity of the most powerful adversary.
- r : the size of the random string R transmitted during a round t.
- b: the size of a block of the random string R transmitted during a round t.
- l: the number of indexes specified by a given document.
- n: the number of blocks of the random string R transmitted during a round t.

We assume that:

(1) $r \gg s$: The size of the random string R transmitted during a round t is greater than the storage capacity of the most powerful user of the system.

(2) $1 < n/|H| \ll l$: The first inequality means that the number of the blocks of R transmitted during a round t is greater than the number of blocks in a sketch saved by a potential user. The second inequality means that there exists an integer $u > 1$ such that $n = u.\,|H|$ with u much smaller than the number of indexes specified by a document.

(3) $2b \gg r/b$: The number of possible values for a block of size b bits is greater than the number of blocks of the string R transmitted during a round t.

(4) $r/b > l$: The number of blocks of the string R transmitted during a round t is greater than the number of shares used in the adopted Shamir's secret sharing scheme.

(5) $b \gg 1$: The size of a block of R is much greater than 1 bit.

In our security study we demonstrate mainly two important characteristics of our non-interactive timestamping scheme. First, we prove that backdating documents in our timestamping scheme has a negligible probability. Second, we prove that the timestamps provided by our timestamping scheme have an eternal validity.

6.1 Negligible Probability of backdating documents

In our timestamping scheme, the probability that an adversary backdates a document D for a string R already transmitted using his stored blocks of R is negligible. This proof is established in two steps. In the first step, we show that if an adversary A wants to backdate successfully a document D for a random string R, then he must backdate it "correctly" for this string R with an error err negligible. In the second step, we show that the probability of backdating correctly a document D for a random string R is negligible.

First step If the adversary produces a timestamp of D such that the vector of blocks associated to $Share(D)$ according to this timestamp is far in Hamming distance from the vector of blocks associated to the timestamp produced by $Stamp(D, Store(D, R))$ then the verifier may with a high probability reject the timestamp of the adversary because the values indexed by $Share(D)$ according to the timestamp do not match the values indexed by $Share(D)$ according to his sketch. Given a correct timestamp T and a timestamp produced by an adversary A for the document D and the random string R denoted T^*, the adversary backdates the document D for R successfully, if he produces a timestamp T^* such that the vector of blocks associated to $Share(D)$ according to T^* denoted ${R^*}_{|Share(D)}$ is close in Hamming distance to $R_{|Share(D)}$. In this case, we say that the adversary backdates "correctly"the document D for the random string R with err errors. Where err is an integer very close to zero. More formally, let A be an adversary. Denote $R_{successful(D)} = {R^\gamma}_{successful(D)}$ the set of strings R for which A has the necessary storage to try to backdate the document D with a probability of success greater than γ.

Lemma 1. *If an adversary backdates a document D for a random string R with a probability of success γ then he backdates it "correctly"with at most $(n/|H|)ln(1/\gamma)$ errors. [16]*

Proof. Let us suppose that the adversary provides a timestamp for the document D for the string R such that the timestamp is made with $err^* > err$ incorrect indices. Denote $INCORRECT(D, R)$ the set of incorrect indices for D and R. If $H \cap INCORRECT(D, R) \neq \emptyset$ the verifier will reject the timestamp of the adversary.

Let i be an indice of H, the probability that i be in $INCORRECT(D, R)$ is : $Pr[i \in INCORRECT(D, R)] = err^*/n$.

The probability that i does not belong to $INCORRECT(D, R)$ is $1 - Pr[i \in INCORRECT(D, R)] = 1 - err^*/n$.

The probability that all the elements of $|H|$ do not belong to $INCORRECT(D, R)$ is:

$Pr[\forall i \in H,\ i \notin INCORRECT(D, R)] = |H|.(1 - err^*/n) \leq e^{-(err^*|H|)/n}$.

If an adversary backdates a document D for a random string R with a probability of success $\gamma \leq e^{-(err^*|H|)/n} \leq e^{-(err|H|)/n}$ then he backdates it correctly with at most $err \leq n/|H|.ln(1/\gamma)$.

Denote $R_{correct}(D)$ the set of strings R for which A can "correctly"backdate D with at most $(n/|H|)ln(1/\gamma)$ errors.

Theorem 1. *If $err \leq n/|H|.ln(1/\gamma)$ then, $R_{successful}(D)$ is a subset of $R_{correct}(D)$. [16]*

Proof. According to the lemma 1, if an adversary backdates a document D for a random string R with a probability of success γ then he backdates it correctly with at most $err \leq n/|H|.ln(1/\gamma)$. So, if a random string R belongs to $R_{successful}(D)$ then it belongs to $R_{correct}(D)$. Thus, $R_{successful}(D)$ *subseteq*

$R_{correct}(D)$. In addition, more the probability of success γ become close to 1, more this error become close to 0. So, successfully backdating means correctly backdating with a "negligible"error. We prove, in the second step, that the probability that the random string R chosen by the adversary to backdate a document D be in $R_{correct}(D)$ is negligible.

Second step We now prove that for an adversary A a document D and a string R: $Pr[R \in R_{correct}(D)]$ is negligible.

Theorem 2. *If $l \gg 1$ and $b \gg 1$ then $Pr[R \in R_{correct}(D)]$ is negligible, with b the size of a block of R.*

Proof. We proved in the first step, that if an adversary backdates a document "successfully", he backdated it "correctly"with at most a negligible error. Then we proved in the second stage that the probability that the string R for which the adversary tries to backdate the document D be in $R_{correct}(D)$ is negligible.

In fact, knowing that the size of blocks of a random string R is b and the number of these blocks is n, the number of possible random strings is $(2^b)^n$.

Moreover, to backdate a document D, the adversary has to create a timestamp T^* for D such that at least $l - err$ blocks of R indexed by D are used to generate T^*.

In other words, he can try to backdate D only for random strings for which he knows the values of at least $l - err$ blocks from the l blocks indexed by D. Thus, since the adversary tries to correctly backdate D with at most err errors, the number of random strings he can use is at most $(2^b)^{n-l}$.

So, the probability that a random string R belongs to $R_{correct}(D)$ is:

$Pr[R \in R_{correct}(D)] \leq (2^b)^{n-l}/(2^b)^n \leq 1/2^{lb}$. Since $l \gg 1$ and $b \gg 1$, this probability is negligible.

6.2 The Eternal Security of our Timestamping Scheme

In [16], Moran proves that in his non-interactive timestamping scheme, an adversary with a storage M can easily backdate $\delta = M/T$ documents by running the timestamping process on some k documents and storing the generated timestamps (each of which has length at most T). However, the probability that an adversary backdates more than k documents is negligible. We show here that, in our timestamping scheme, after the transmission of R, it is very difficult to forge a fake timestamp for a given document. Moreover, we show that an adversary having a document D and δ correct timestamps can forge a fake timestamp for D only with a negligible probability. Thus, we prove the following theorem:

Theorem 3. *If $2^b \gg r/b$ and $r/b > l$ then the probability to forge a fake timestamp for a document D and a string R using δ correct timestamps related to R is negligible.*

Proof. The inequality $2^b \gg r/b$ means that the number of values for a block of size b bits is greater than the number of blocks of the string R transmitted during a round t.

l is the number of indices specified by a given document, this number must always be less than the number of blocks of R. So, $r/b > l$.

It follows that the $2^b \gg l$, which means that the number of possible values for a block of R is much greater than the number of indices specified by the document D.

For each timestamp T_j $(1 \leq j \leq \delta)$, if the adversary gives any value v from the 2^b possible values of a block of R, it will recover a given index i.

However, the fact that $i = T_j(v)$ does not mean that $R_i = v$. This means that i is the value associated to v by the polynomial T_j but the couple (i, v) does not necessarily belong to the string R. In other words, the string R may not associate the value v to the block indexed by i. Moreover, it may exist i such that $R_{i'} = v$ and there is no way to verify if $i = i'$. The only points of T_j for which the adversary knows that they belong to R are the points whose indices are specified by the document associated to T_j. The probability that the adversary chooses one of these points is $l/2^b \ll 1$.

To obtain the k points required to forge a fake timestamp, the probability is negligible since it is the product of the probabilities of selecting each of the points belonging to a valid timestamp.

Thus, the adversary can obtain the k points needed to forge a fake timestamp for a document D for a random string R with a negligible probability.

7 Conclusion

In this paper, we have presented a non-interactive timestamping solution in the bounded storage model. Our solution is not interactive and hides even the fact that a timestamping occurred. It also ensures total confidentiality of the provided timestamps. In addition, our solution provides eternal security for the provided timestamps. In fact, neither increasing the storage capacity of an adversary or the evolution of his computing power will compromise a provided timestamp. Thus, our solution is more secure than existing systems whose timestamps can be challenged when the computing power or storage capacity of users increase. In this context, we studied the security of our solution and formally proved that the possibility of cheating is negligible. In our future work, we plan to adopt new secret sharing schemes for timestamping.

References

1. Y. Aumann and Y. Z.Ding and M. O. Rabin, *Everlasting security in the bounded storage model*, IEEE Transactions on Information Theory. 2002.
2. D. Bayer and S. Haber and W. S. Stornetta, *Improving the efficiency and reliability of digital timestamping*, In: Sequences91: Methods in Communication, Security and Computer Science. 1992.

3. A. Ben Shil and K. Blibech and R. Robbana, *A New Timestamping Schema in the Bounded Storage Model*, In: Proceedings of the 3rd Conference on Risks and Security of Internet and Systems. CRiSIS. 2008.
4. K. Blibech and A. Gabillon, *A New Timestamping Scheme Based on Skip Lists*, In: ICCSA (3). pp. 395-405, 2006.
5. K. Blibech and A. Gabillon, *A New Totally Ordered Timestamping Scheme*, In: 5th Conference on Security and Network Architectures SAR. . 2006.
6. K. Blibech and A. Gabillon, *CHRONOS: an authenticated dictionary based on skip lists for timestamping systems*, In: SWS. pp. 84-90, 2005.
7. K. Blibech and A. Gabillon and A. Bonnecaze, *Etude des systèmes d'horodatage*, Technique et Science Informatiques 26(3-4). pp. 249-278, 2007.
8. A. Bonnecaze and P. Liardet and A. Gabillon and K. Blibech, *A distributed time stamping scheme*, 4th Conference on Security and Network Architectures SAR 2005. 2005.
9. A. Budas and P. R. Laud and H. Lipmaa and J. Willemson, *Timestamping with Binary Linking Schemes*, In: CRYPTO. ICICS. pp. 486-501, 1998.
10. A. Budas and P. R. Laud and B. Schoenmakers, *Optimally efficient accountable timestamping*, In: Public Key Cryptography. 2000.
11. S. Dziembowski and U. Maurer, *On Generating the Initial Key in the Bounded-Storage Model*. In: Advances in Cryptology- EUROCRYPT. pp.126-137, 2004.8
12. S. Haber and W. S. Stornetta, *How to Time-Stamp a Digital Document*, J. Cryptology 3(2). pp. 99-111, 1991.
13. U. Maurer, *Conditionally-perfect secrecy and a provably-secure randomized cipher*, Journal of Cryptology. pp. 53-66, 1992.
14. U. Maurer, *Secret key agreement by public discussion*, IEEE Transaction on Information Theory, 39. pp. 733-742, 1993.
15. T. Moran and R. Shaltiel and A. Ta-Shma, *Non-interactive Timestamping in the Bounded Storage Model*, In: In Advances in Cryptology. CRYPTO'04. LNCS, Springer, 3152. pp. 460-476, 2004.
16. T. Moran and R. Shaltiel and A. Ta-Shma, *Non-interactive Timestamping in the Bounded Storage Model*, In: J. Cryptology. pp. 189-226, 2009.
17. National Institute of Standards and Technology (NIST), *Announcement of Weakness in the Secure Hash Standard*, Technical report. 1994.
18. A. Shamir, *How to share a secret*, Commun. ACM 22(11). pp. 612-613, 1979.