Universidade Técnica de Lisboa

Instituto Superior Técnico

Mathematics Department

Quantitative Analysis of Security Protocols

Tiago Monteiro Grilo dos Reis

Diploma Thesis

Licenciatura em Matemática Aplicada e Computação

Supervisor:

Prof. Paulo Mateus

October 2005

Acknowledgements

I would like to thank my supervisor, Professor Paulo Mateus, for his guidance, support and constant motivation throughout this year and especially on the final stage of this work.

I would also like to thank Professor Luca Viganò for his support and for his talks here in Lisbon that provided the first interest for Security Protocol Model Checking and inspiration for this work.

I want to thank the CLC (Center for Logic and Computation, Instituto Superior Técnico) for taking me as a student member on the course of this year and sponsoring my trip to ETH Zurich.

This work was partially supported by FCT and FEDER through POCTI and POCI, namely via QuantLog POCI/MAT/55796/2004 Project of CLC.

Contents

Chapter 1

Introduction

In the last few years, several different tools for formal analysis of security protocols have been proposed. Most of these tools are based in model checking. Despite the increasing success of these tools, many protocols have been proved to be correct, several other protocols have not been analyzed and seem to be out of scope for these provers. The reason being is that these tools reflect an implementation of the well know Dolev-Yao model [**?**] and this model is too weak to specify several types of security protocols, namely those including coin tossing, and intrinsic cryptographic primitives, some examples are: zero-knowledge proof system; oblivious transfer; secure computation; bit commitment.

Moreover, it is well known that protocols can be prove secure for the Dolev-Yao model, but insecure when considering specific cryptosystems. In [**?**], it is shown that a "correct" implementation of the NSL authentication protocol [**?**] can be attacked if El-Gamal is used as the underlying cryptosystem.

This problem arises because this verification of protocols is done in a purely syntactic (or symbolic) way and problems with the real implementation of the protocols are not addressed by these model-checkers. Even when using results that relate symbolic and complexity models [**?**, **?**], the equivalence is up to a negligible function and so attacks might be found when we are in the "real world".

In this work we intend to bridge a little bit further the gap between formal verification and concrete implementations by introducing an extension of the usual Dolev-Yao model, we will call it $\mathcal{DY}^{+}$. This new adversary is able to perform attacks such as guessing encryption keys, we allow this because under very relaxed assumptions, when we take in account specific implementations of security protocols some attacks are possible for an attacker with a reasonable amount of computational power.

We will do this by introducing new derivation rules which are dependent of the encryption scheme that is being used. Until now, encryption was compared to a "black box" operation that could only be undo with the right decryption key,

in our work more structure must be associated to each encryption, such as key space. We will also expand the model-checker so that each branch in the states' tree is weighed with the probability associated with that transition. For instance, a state where it is possible to guess a key will have a successor that includes the knowledge of that key, but this transition will be weighed with the probability of guessing the key.

Since each transition of the states' tree will be weighed with the probability of that transition, in the end we will be able to tell with which probability each attack is possible.

The classical Dolev-Yao intruder is limited by the assumption of perfect cryptography, we aim in augmenting the classic intruder model with the power of being able to break some encryption, given a certain probability.

Our work is based on model checking tool for security protocols that is part of the AVISPA project. The research conducted under the AVISPA project lead to the development of an automated tool for the verification of security protocols, this tool can currently handle industrial-complexity protocols. The AVISPA tool is a collection of four protocol verifiers, we base this work in one of them, the *on-the-fly-model-checker* that is explained in [**?**], we will start by recalling some key concepts present in [**?**], present the classical intruder and then suggest an extension. Finally we will modify the key concepts in order to deal with the augmented Dolev-Yao intruder.

Chapter 2

Previous Work

In this chapter we present an overview of the model checking tool for analyzing security protocols *OFMC*, the *on-the-fly-model-checker*. We cover the protocol specification languages, the intruder knowledge generation rules and the semantics that lead to the construction of the infinite state transition system.

2.1 Syntax of the Intermediate Format

Two languages for protocol specification are available. The *High Level Protocol Specification Language* was created as a high level language to serve as the interface language between the user and the model checking tool, it is the input language for the user to write down the protocol specification like he was using *Alice&Bob* notation.

Internally the model checking tool does not work the high level language, for the automated analysis a low level language is used. The transition system states are represented by the Intermediate Format *IF*. The *IF* language is hidden from the user, all the input provided in *HLPSL* is automatically translated to the *IF*.

We will be dealing directly with the *IF* language throughout our work because the *Intermediate Format* is the language of the model checking tool.

Definition 1 (*IF* Language)**.** Let $\mathcal{C}$ and $\mathcal{V}$ be disjoint countable sets of *constants* (denoted by lowercase letters) and *variables* (denoted by uppercase letters). The

syntax of the *IF* language is defined by the following context-free grammar:

$$
\begin{aligned}
ProtocolDescr &::= (State, Rule^*, AttackRule^*)\\
Rule &::= LHS \Rightarrow RHS\\
AttackRule &::= LHS\\
LHS &::= State\ NegFact\ Condition\\
RHS &::= State\\
State &::= PosFact(.PosFact)^*\\
NegFact &::= (.\mathsf{not}(PosFact))^*\\
PosFact &::= \mathsf{state}(Msg) \mid \mathsf{msg}(Msg) \mid \mathsf{i\_knows}(Msg) \mid \mathsf{secret}(Msg, Msg)\\
Condition &::= (\wedge\ Msg \neq Msg)^*\\
Msg &::= AtomicMsg \mid ComposedMsg\\
ComposedMsg &::= \langle Msg, Msg\rangle \mid \{Msg\}_{Msg} \mid \{|Msg|\}_{Msg} \mid Msg(Msg) \mid Msg^{-1}\\
AtomicMsg &::= \mathcal{C} \mid \mathcal{V} \mid \mathbb{N} \mid \mathsf{fresh}(\mathcal{C}, \mathbb{N})
\end{aligned}
$$

We write $\mathcal{L}(n)$ for the context-free language associated with the nonterminal n. We write $vars(t)$ to denote the set of variables occurring in a (message, fact, or state) *term* t, and when $vars(t) = \emptyset$, we say that t is *ground*, and write $ground(t)$. We straightforwardly extend the functions *vars* and *ground* to the more complex terms.

Notation 1. We denote *IF* constants with lowercase sans-serif, *IF* variables with uppercase sans-serif, meta-variables (i.e., variables ranging over message terms) with *lowercase italics*, and sets with *uppercase italics*.

The language fragment associated to messages $\mathcal{L}(Msg)$ plays an important role in defining the intruder model further ahead. Messages can be either atomic or composed. For atomic messages we allow constants, variables, natural numbers and one time use constants that will represent unique data, such as randomly generated natural numbers or any other sort of data. Messages can be composed using pairing $\langle m_1, m_2\rangle$, or any of the following cryptographic primitives: symmetric encryption $\{|m_1|\}_{m_2}$, asymmetric encryption $\{m_1\}_{m_2}$, application of a function $f(m)$ usually for hash function modeling or key-table lookup and lastly asymmetric inverse m^{-1}.

Definition 2 (Substitution)**.** A *substitution* σ is a mapping from $\mathcal{V}$ to $\mathcal{L}(Msg)$. The *domain* of σ, denoted by $dom(\sigma)$, is the set of variables $V \in \mathcal{V}$ such that $\sigma(v) \neq v$ iff $v \in V$. The *co-domain* of σ, denoted by $co\text{-}dom(\sigma)$ is the set $\{\sigma(v) | v \in dom(\sigma)\}$. As we only consider substitutions with finite domains, we represent a substitution σ with $dom(\sigma) = \{v_1, \ldots, v_n\}$ by $[v_1 \mapsto \sigma(v_1), \ldots, v_n \mapsto \sigma(v_n)]$. The *identity substitution id* is the substitution with $dom(id) = \emptyset$. We say that a substitution σ is *ground*, and write $ground(\sigma)$, if $\sigma(v)$ is a ground term for all

$v \in dom(\sigma)$. We extend σ to a homomorphism on message terms, facts, and states in the standard way, and we also write $t\sigma$ as a shorthand for $\sigma(t)$.

We say that two substitutions σ_1 and σ_2 are *compatible*, written $\sigma_1 \approx \sigma_2$, if $v\sigma_1 = v\sigma_2$ for every $v \in dom(\sigma_1) \cap dom(\sigma_2)$. The *composition* of σ_1 and σ_2 is denoted by $\sigma_1\sigma_2$. Note that $\sigma_1\sigma_2 = \sigma_2\sigma_1$ for compatible ground substitutions. For two sets of ground substitutions Σ_1 and $Sigma_2$, we define their *intersection modulo the different domains* as

$$\Sigma_1 \sqcap \Sigma_2 = \{\sigma_1\sigma_2 | \sigma_1 \in \Sigma_1 \wedge \sigma_2 \in \Sigma_2 \wedge \sigma_1 \approx \sigma_2\}.$$

Since the composition of compatible ground substitutions is associative and commutative, so is the $\sqcap$ operator.

Two terms *unify* when there exists a substitution, called their *unifier*, under which they are equal. *Matching* is the special case where one of the terms is ground. Since we are working under the free algebra assumption, two unifiable terms always have a *most general unifier* (mgu).

Finally, for ϕ a propositional combination of equalities and for σ a substitution for the free variables of ϕ, we define the relation $\sigma \vDash \phi$ to represent that ϕ is satisfied by σ in the structure given by the freely generated term algebra (in our case with the carrier set $\mathcal{L}(Msg)$).

To specify state transitions and identification of attack states we use rules with the form $LHS \Rightarrow RHS$. Transition rules correspond to steps of the protocol between honest agents, the attack rules are not related to protocol steps, they are created depending on the security property we are checking. Both sides of the rule may contain variables and we require that $vars(LHS) \subseteq vars(RHS)$. We just consider rules of the form

$$\mathsf{msg}(m_1).\mathsf{state}(m_2).P_1.N_1 \wedge \mathrm{Cond} \Rightarrow \mathsf{state}(m_3)\mathsf{msg}(m_4).P_2.$$

The *LHS* is a set of positive facts P, a set of negative facts N and a condition *Cond*, that satisfy $\mathrm{vars}(Cond) \cup \mathrm{vars}(N) \subseteq \mathrm{vars}(P)$. In the *RHS* is a set of positive facts. The applicability of a rule to a state is as follows.

Definition 3. A rule $LHS \Rightarrow RHS$ is applicable to a state S if

1. The positive facts are contained in the state for some substitution σ of the rule's variables;
2. the negative facts under σ are not contained in the state;
3. the condition is satisfied under σ.

A protocol is described by a triple (I, R, AR), where I is the initial *state*, R is a set of *rules* and AR is an *attack-rule*. We require that the initial state be a ground term. Further ahead we refer to the protocol descriptions $\mathcal{L}(ProtocolDescr)$ to introduce the semantics of the *IF*.

2.2 The Dolev-Yao Intruder

We turn now to the classical attacker model, the Dolev-Yao intruder. Roughly, the Dolev-Yao intruder is an agent that completely dominates the network but cannot break cryptography. The intruder has total control of the information passing on the network, it can personify other agents also using the network, it can prevent sent messages from reaching their destination or reroute them to other agents.

The attacker can manipulate message flow at will and send any message it can generate from the knowledge it acquires from the network, but cannot break the encryption of the messages it intercepts unless it has the corresponding key. The intruder agent is often referred as the malicious or dishonest agent, and the protocol participants are referred as the honest agents.

An important part of the intruder model is the intruder knowledge. The attacker begins with a small set of facts but as it captures messages from the network its knowledge expands. With the increase on the number of message more terms can be generated by combining the intruder knowledge and thus providing the attacker better chances at succeeding the attack.

Definition 4. For a set M of messages, let the intruder knowledge $\mathcal{DY}(M)$ (for Dolev-Yao) be the smallest set closed under the following generation (G) and analysis (A) rules:

$$\frac{m \in M}{m \in \mathcal{DY}(M)}\ (G_{\text{AXIOM}})$$

$$\frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{\langle m_1, m_2 \rangle \in \mathcal{DY}(M)}\ (G_{\text{PAIR}})$$

$$\frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{\{m_2\}_{m_1} \in \mathcal{DY}(M)}\ (G_{\text{CRYPT}})$$

$$\frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{\{\!|m_2|\!\}_{m_1} \in \mathcal{DY}(M)}\ (G_{\text{SCRYPT}})$$

$$\frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{m_1(m_2) \in \mathcal{DY}(M)}\ (G_{\text{APPLY}})$$

$$\frac{\langle m_1, m_2\rangle \in \mathcal{DY}(M)}{m_i \in \mathcal{DY}(M)}\ (A_{\text{PAIR}})$$

$$\frac{\{m_2\}_{m_1} \in \mathcal{DY}(M) \quad m_1^{-1} \in \mathcal{DY}(M)}{m_2 \in \mathcal{DY}(M)}\ (A_{\text{CRYPT}})$$

$$\frac{\{|m_2|\}_{m_1} \in \mathcal{DY}(M) \quad m_1^{-1} \in \mathcal{DY}(M)}{m_2 \in \mathcal{DY}(M)}\ (A_{\text{SCRYPT}})$$

$$\frac{\{m_2\}_{m_1^{-1}} \in \mathcal{DY}(M) \quad m_1 \in \mathcal{DY}(M)}{m_2 \in \mathcal{DY}(M)}\ (A^{-1}_{\text{CRYPT}})$$

The generation rules express that the intruder can compose messages from known messages using pairing, asymmetric and symmetric encryption, and function application. The analysis rules describe how the intruder can decompose messages. Note that no rules are given that allow the intruder to analyze function applications, for example to recover m from $f(m)$. Moreover, note that this formalization correctly handles non atomic keys, for instance $m \in \mathcal{DY}(\{\{|m|\}_{f(k_1,k_2)}, k_1, k_2, f\})$.

2.3 The Semantics of the Intermediate Format

Each protocol is represented by a triple (I, R, AR), where I is the initial state, R is the set of transition rules and AR a rule for identifying an attack state. Protocols are modeled by an infinite-state transition system. We build such a transition system starting with the initial state I containing the initial intruder knowledge and the knowledge of the honest protocol agents, from there, each successor state results from unifying the *lhs* of the transition rules in R using only admissible substitutions.

Definition 5. Let $r = lhs \Rightarrow rhs$ be a rule of the form

$$\mathsf{msg}(m_1).\mathsf{state}(m_2).P_1.N_1 \wedge \text{Cond} \Rightarrow \mathsf{state}(m_3)\mathsf{msg}(m_4).P_2,$$

and let $\overline{P_1}$ be obtained from P_1 by removing all $\mathsf{i\_knows}$ facts,

$$\overline{P_1} = P_1\{f | \exists m.f = \mathsf{i\_knows}(m)\}.$$

we define the applicability of such a rule r by the function applicable that maps a state S and the left-hand side *lhs* of r to the set of ground substitutions under which the rule can be applied to the state:

$$\begin{aligned} \mathit{applicable}_{lhs}(S) = \{\sigma| \\ & \mathrm{ground}(\sigma) \wedge \mathrm{dom}(\sigma) = \mathrm{vars}(m_1) \cup \mathrm{vars}(m_2) \cup \mathrm{vars}(P_1) \\ & \wedge \{m_1\sigma\} \cup \{m\sigma | \mathsf{i\_knows}(m) \in P_1\} \subseteq \mathcal{DY}(\{m | \mathsf{i\_knows}(m) \in S\}) \\ & \wedge \mathsf{state}(m_2\sigma) \in S \wedge \overline{P_1}\sigma \subseteq \\ & \wedge (\forall f.not(f) \in N_1 \Longrightarrow f\sigma \notin S) \wedge \sigma \vdash \mathit{Cond}\}. \end{aligned}$$

We define the successor function

$$\mathit{succ}_R(S) = \bigcup_{r \in R} \mathit{step}_r(S)$$

that given a set R of rules of the above form and a state S yields the corresponding set of successor states by means of the following step function:

$$\begin{aligned} \mathit{step}_{lhs \Rightarrow rhs}(S) = \{S' | \exists \sigma. \\ & \sigma \in \mathit{applicable}_{lhs}(S) \\ & \wedge S' = (S \backslash (\mathsf{state}(m_2\sigma) \cup \overline{P_1}\sigma)) \cup \mathsf{i\_knows}(m_4\sigma) \cup P_2\sigma\}. \end{aligned}$$

The transition system that models a protocol (I, R, AR) is built by applying the successor function to the initial state I of the protocol. With this process we obtain the set of reachable states, and it is important to note that this set is a *ground model*. No reachable state contains variables because of the way we define the transition.

The set of reachable states is the defined as follows.

Definition 6. The set of reachable states of the protocol (I, R, AR) is the set

$$\mathit{reach}(I, R) = \bigcup_{n \in \mathbb{N}} \mathit{succ}_R^n(I).$$

Definition 7. Given an attack rule AR and a state S, the attack predicate $\mathit{isAttack}_{AR}(S)$ is true iff the rule AR can be applied to the state S, that is $\mathit{applicable}_{AR}(S) \neq \emptyset$.

2.4 Complexity results

In this section we analyze the complexity of the model-checking technique described in the previous sections. The main result is that under the assumption

of finite number of session the problem of checking whether a state holding an attack is reachable or not is *NP*-complete, therefore checking safety properties will be Co-*NP* complete. We start by presenting the decision problem for which the complexity result holds.

Definition 8 (Attack problem for k sessions)**.** Given a protocol and an attack rule (I, R, AR), determine whether this protocol has an attack where k session are run.

As promised, the next result determines that this problem is *NP*-complete.

Proposition 1. The attack problem for k sessions is *NP*-complete (where the complexity is measured on the state representation of the protocol).

Proof. We start by proving the the problem is *NP*. For that it is enough to show that we can verify an attack in polynomial time. This is straightforward since it resumes to provide a trace of the attack and the corresponding rules, which are polynomial on the size of the protocol representation, and note that we only consider protocols with a constant number of steps.

Therefore, it remains to show that the problem is *NP*-hard. For this end we will reduce the *SAT* to finding an attack. Consider the following toy protocol: Alice A knows a set Φ of propositional symbols (which we assume to be totally ordered), Bob B has a formula φ over this propositional symbols and a secret s. The protocols has $|\Phi|$ steps where in step n, Alice sends a bit to Bob corresponding to a value of the n-th propositional symbol. At the end of the protocol B has a valuation for φ and he sends the secret to A in a public channel if the valuation satisfies φ. Therefore, since the attacker has Byzantine power (that is, he can change the values of the bits sent by A), if the attacker changes the bits correctly he will know the secret. That is, finding the secret is essentially finding a valuation that satisfies φ. □

Since, the attack problem is *NP*-complete, and at the moment we expect $P \neq NP$, the model checking algorithm is exponential in the representation of the protocol.

Chapter 3

Extending the Adversary Model

The classical Dolev-Yao intruder assumes perfect cryptography and can only decrypt a message if it knows the corresponding encryption key. We aim to augment the classical adversary model in such a way it allows the intruder to perform attacks on the cryptography, by doing such we can no longer ignore the underlying cryptographic schemes used in the protocol specification.

The attacker is allowed to handle an encrypted message and retrieve its encryption key, but only with some probability of success. The intruder can then generate and analyze protocol messages using this new piece of information possibly resulting on a significant increase of the intruder knowledge opening new ways for attacks. It is important to accurately measure the probability of correctly retrieving the encryption key, either by simple guessing or resorting to cryptanalysis techniques.

With this new adversary model we expect to detect flaws in protocols caused by badly chosen cryptographic systems in important positions of the protocol flow, such as repetitive use of the same encryption key that allows the intruder to gather enough information to be able to compute the key with a high probability of success or easily guessable passwords that compromise the security of the protocol.

3.1 Quantitative Analysis

We decided to empower our adversary with a key retrieving ability because cryptanalysis is possible. In some cases brute force attacks are feasible and when the security relies on passwords pure guessing or dictionary attacks work with a high probability of success, high enough to present a security problem.

To quantify the effort needed to break message encryption we follow an approach based on the dimensions of the key space of a given cryptographic system taking into account the fact that additional knowledge may reduce the key space

significantly.

For this we assume a function $p : \mathcal{L}(Msg) \times \mathcal{P}(\mathcal{L}(PosFact)) \to \mathbb{R}$ to exist.

It returns the probability of guessing the encryption key $k \in \mathcal{L}(Msg)$ when the intruder knowledge is IK, a set of positive facts of the form $\mathsf{i\_knows}(Msg)$. The probability of success possibly depends on the current knowledge of the intruder.

Other approaches could be used, like the amount of computational power needed for a brute force attack. Or the likelihood of a successful attack given a time bound for cryptanalysis. Of course these approaches are highly dependent of the assumptions on the computational power of the intruder, nonetheless they can be tuned to mirror the capabilities of a relatively powerful intruder.

3.2 Changes on the Syntax of the *IF*

Minor changes on the syntax are needed to accommodate the new information being introduced at each state, that is, the probability of a transition when a key is guessed by the intruder. The *IF* languages is as before with the added unary operator plabel as follows.

Definition 9 (IF^+ Language)**.** Let $\mathcal{C}$ and $\mathcal{V}$ be disjoint countable sets of *constants* (denoted by lowercase letters) and *variables* (denoted by uppercase letters). The syntax of the *IF* language is defined by the following context-free grammar:

$$
\begin{aligned}
ProtocolDescr &::= (State, Rule^*, AttackRule^*) \\
Rule &::= LHS \Rightarrow RHS \\
AttackRule &::= LHS \\
LHS &::= State\ NegFact\ Condition \\
RHS &::= State \\
State &::= PosFact(.PosFact)^* \\
NegFact &::= (.\mathsf{not}(PosFact))^* \\
PosFact &::= \mathsf{state}(Msg) \mid \mathsf{msg}(Msg) \mid \mathsf{i\_knows}(Msg) \mid \mathsf{secret}(Msg, Msg) \mid \mathsf{plabel}(\mathbb{R}) \\
Condition &::= (\wedge\ Msg \neq Msg)^* \\
Msg &::= AtomicMsg \mid ComposedMsg \\
ComposedMsg &::= \langle Msg, Msg \rangle \mid \{Msg\}_{Msg} \mid \{|Msg|\}_{Msg} \mid Msg(Msg) \mid Msg^{-1} \\
AtomicMsg &::= \mathcal{C} \mid \mathcal{V} \mid \mathbb{N} \mid \mathsf{fresh}(\mathcal{C}, \mathbb{N})
\end{aligned}
$$

3.3 The Extended Dovel-Yao Intruder

The way the intruder extends its knowledge needs also to be reviewed to reflect the new kind of attack. Essentially the way the intruder reasons will only be affected by the new key guessing ability. We maintain the same set of rules as before and add two guessing rules for symmetric and public key encryption.

Definition 10. For a set M of messages let $\mathcal{DY}^+(M)$ be the smallest set closed under the following generation (G) and analysis (A) rules plus two rules for guessing symmetric and public keys GUESSS and GUESSPK.

$$\frac{m \in M}{m \in \mathcal{DY}^+(M)}\ (G_{\text{AXIOM}})$$

$$\frac{m_1 \in \mathcal{DY}^+(M) \quad m_2 \in \mathcal{DY}^+(M)}{\langle m_1, m_2 \rangle \in \mathcal{DY}^+(M)}\ (G_{\text{PAIR}})$$

$$\frac{m_1 \in \mathcal{DY}^+(M) \quad m_2 \in \mathcal{DY}^+(M)}{\{m_2\}_{m_1} \in \mathcal{DY}^+(M)}\ (G_{\text{CRYPT}})$$

$$\frac{m_1 \in \mathcal{DY}^+(M) \quad m_2 \in \mathcal{DY}^+(M)}{\{\!|m_2|\!\}_{m_1} \in \mathcal{DY}^+(M)}\ (G_{\text{SCRYPT}})$$

$$\frac{m_1 \in \mathcal{DY}^+(M) \quad m_2 \in \mathcal{DY}^+(M)}{m_1(m_2) \in \mathcal{DY}^+(M)}\ (G_{\text{APPLY}})$$

$$\frac{\langle m_1, m_2 \rangle \in \mathcal{DY}^+(M)}{m_i \in \mathcal{DY}^+(M)}\ (A_{\text{PAIR}})$$

$$\frac{\{m_2\}_{m_1} \in \mathcal{DY}^+(M) \quad m_1^{-1} \in \mathcal{DY}^+(M)}{m_2 \in \mathcal{DY}^+(M)}\ (A_{\text{CRYPT}})$$

$$\frac{\{\!|m_2|\!\}_{m_1} \in \mathcal{DY}^+(M) \quad m_1^{-1} \in \mathcal{DY}^+(M)}{m_2 \in \mathcal{DY}^+(M)}\ (A_{\text{SCRYPT}})$$

$$\frac{\{m_2\}_{m_1^{-1}} \in \mathcal{DY}^+(M) \quad m_1 \in \mathcal{DY}^+(M)}{m_2 \in \mathcal{DY}^+(M)}\ (A^{-1}_{\text{CRYPT}})$$

$$\frac{\{m_2\}_{m_1} \in \mathcal{DY}^+(M) \quad m_1 \notin \mathcal{DY}^+(M)}{m_1 \in \mathcal{DY}^+(M)}\ (\text{GUESSS})$$

$$\frac{\{\!|m_2|\!\}_{m_1} \in \mathcal{DY}^+(M) \quad m_1^{-1} \notin \mathcal{DY}^+(M)}{m_1^{-1} \in \mathcal{DY}^+(M)}\ (\text{GUESSPK})$$

The intruder can still use pairing, symmetric and asymmetric encryption to compose and decompose messages. We add the possibility of the attacker breaking a symmetric and asymmetric encryption and gain knowledge of the encrypted message together with the encryption key.

The new set of messages $\mathcal{DY}^+(M)$ obviously contain more messages with this new definition than the previous intruder knowledge $\mathcal{DY}(M)$, it's evident that $\mathcal{DY}(M) \subseteq \mathcal{DY}^+(M)$.

3.4 A New Semantics for the Intermediate Format

To incorporate the changes in the adversary model, namely the new intruder knowledge definition, the way the transition system is build needs to be modified. Every state transition must reflect the fact that the intruder may have used the guessing rule, thus, we need to distinguish the deterministic transitions from the probabilistic ones. To achieve this distinction we label the transition with the value we get from the function p. If no guessing was needed p returns 1. When a state is found that triggers the attack rule, we determine the attack probability by backtracking the attack trace and multiply all the probability labels. We only introduce new attacks, the old ones will be detected and will be labeled with probability one.

Definition 11. Let $r = lhs \Rightarrow rhs$ be a rule of the form

$$\mathsf{msg}(m_1).\mathsf{state}(m_2).P_1.N_1 \wedge \mathrm{Cond} \Rightarrow \mathsf{state}(m_3)\mathsf{msg}(m_4).P_2,$$

and let $\overline{P_1}$ be obtained from P_1 by removing all $\mathsf{i\_knows}$ facts,

$$\overline{P_1} = P_1\{f|\exists m.f = \mathsf{i\_knows}(m)\}.$$

we define the applicability of such a rule r by the function applicable that maps a state S and the left-hand side lhs of r to the pair $\langle\sigma, \xi\rangle$, where σ is a ground substitution and $\xi \in \mathbb{R}$, under which the rule can be applied to the state:

$$\begin{aligned}
applicable_{lhs}(S) = \{\langle\sigma,\xi\rangle| & && (3.1)\\
& \mathrm{ground}(\sigma) \wedge \mathrm{dom}(\sigma) = \mathrm{vars}(m_1) \cup \mathrm{vars}(m_2) \cup \mathrm{vars}(P_1) && (3.2)\\
& \wedge \{m_1\sigma\} \cup \{m\sigma|\mathsf{i_knows}(m) \in P_1\} \subseteq \mathcal{DY}^+(\{m|\mathsf{i_knows}(m) \in S\}) && (3.3)\\
& \wedge \xi = \Pi_{t\in \text{co-dom}(\sigma)} p(t, \mathcal{DY}^+(\{m|\mathsf{i_knows}(m) \in S\})) && (3.4)\\
& \wedge \mathsf{state}(m_2\sigma) \in S \wedge \overline{P_1}\sigma \subseteq && (3.5)\\
& \wedge (\forall f.not(f) \in N_1 \Longrightarrow f\sigma \notin S) \wedge \sigma \vdash Cond\}. && (3.6)
\end{aligned}$$

We define the successor function

$$succ_R(S) = \bigcup_{r \in R} step_r(S)$$

that given a set R of rules of the above form and a state S yields the corresponding set of successor states by means of the following step function:

$$step_{lhs \Rightarrow rhs}(S) = \{S' | \exists \sigma. \tag{3.7}$$
$$\langle \sigma, \xi \rangle \in applicable_{lhs}(S) \tag{3.8}$$
$$\wedge S' = (S \backslash (\mathsf{state}(m_2\sigma) \cup \overline{P_1}\sigma)) \cup \mathsf{i\_knows}(m_4\sigma) \cup P_2\sigma \cup \mathsf{plabel}(\xi)\}. \tag{3.9}$$

Definition 12. Given an attack-predicate AR, if $isAttack_{AR}(S)$ is true, then the probability of such an attack $pAttack_{AR}(S)$ is given by

$$pAttack_{AR}(S) = \prod_{x \in \{t | \mathsf{plabel}(t) \in S\}} x.$$

3.5 Complexity results

In this section we analyze the complexity of the model-checking technique for the probabilistic extension.

The extension we propose does not change the model checking procedure, apart from the probability labels introduced in the transition system states.

Proposition 2. The attack problem for k sessions is NP-complete.

Proof. Follow the proof in Proposition **??**. □

3.6 Dependence on the security parameter

The extension we propose requires a clear commitment to the cryptosystem being used in the specified protocol and the security parameter, that is, the size of the keys and the key space. This information is very important because it is required in the calculation of the transition probabilities and attack probability as well. With this model-checking procedure and a correct calculation of probabilities we can use the tool to find out if the probability of the attacks is below a given ϵ, we abandon state exploration when the product of the probability labels of a certain state is less than that given ϵ.

Chapter 4

Conclusion and Future work

We propose an extension on the Dolev-Yao intruder used in the *OFMC* introducing some ideas from the computational approach. Our work is also an attempt to incorporate in security protocol model checkers some probabilistic notions. We introduce some notion of probability in key guessing based on the key space, and under the assumption that the more the intruder knows about the key, the more he restricts the search space and guessing with a significant probability of success is feasible. In this work we lack an appropriate approach on the calculation of the probabilities associated to each encryption key, for we only admit the existence of a function that returns the guessing probability. Further work in this direction is required, the probability calculation must be supported by cryptanalysis results we currently lack and the way the intruder knowledge increases with the new attack rules we introduced must be taken in account.

We also hint alternative ways of quantifying the guessing operation, this topic needs to be carefully addressed to find relevant and realistic ways of measuring feasible cryptanalysis techniques.

The new definitions we introduced, like the changes in the *IF* language and the semantics require a change in the model checking tool in order to proceed to the protocol analysis.

In this work we pointed out complexity results for the problem of finding an attack on a protocol with the model-checking tool. We suggest an attack search up to a given probability of success.

Bibliography

[1] M. Abadi and P. Rogaway, Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption), *Journal of Cryptology* 15, 2 (2002), 103-127.

[2] D. Basin, S. Mödersheim, L. Viganò, OFMC: A symbolic model checker for security protocols, *International Journal of Information Security* 4(3):181–208, 2005.

[3] D. Dolev and A. Yao, On the security of public-key protocols, *IEEE Transaction on Information Theory* 29, 198–208, 1983.

[4] B. Warinschi, A Computational Analysis of the Needham-Schroeder(-Lowe) protocol, *Proceedings of 16th Computer Science Foundation Workshop*, 248–262, 2003.

[5] G. Lowe, An attack on the Needham-Schröder public-key authentication protocol, *Information Processing Letters* 56(3):131–135, 1995.

[6] P. Adão, G. Bana, A. Scedrov, Computational and information-theoretic soundness and completeness of formal encryption, *Proceedings of the 18th IEEE Computer Security Foundations Workshop (CSFW)*,170–184, 2005.

[7] The *AVISPA* Project — Automated Validation of Internet Security Protocols and Applications. `http://www.avispa-project.org/`