Towards a Quantitative Analysis of Security Protocols

Pedro Adão[1], Paulo Mateus[1], Tiago Reis[1], Luca Viganò[2]

[1] CLC, Department of Mathematics, IST, Lisbon, Portugal
[2] Department of Computer Science, ETH Zurich, Switzerland

1 Introduction

In the last few years, a number of tools have been proposed for the formal analysis of security protocols, e.g. [3, 5, 6, 9, 10, 12, 13, 14]). These tools have helped prove several protocols correct and have uncovered flaws in several other protocols. However, there are many other protocols that have not yet been analyzed and which, in fact, are out of the scope of these tools. The main reason for this is that the tools analyze protocols under the assumptions of *perfect cryptography* and that the protocol messages are exchanged over a network that is under the control of a *Dolev-Yao ($\mathcal{DY}$) intruder* [7]. That is, protocols are analyzed by considering the standard protocol-independent, asynchronous model of an active intruder who controls the network but cannot break cryptography; in particular, the intruder can intercept messages and analyze them if he possesses the corresponding keys for decryption, and he can generate messages from his knowledge and send them under any agent name. Hence, the $\mathcal{DY}$ model is too weak to specify several types of security protocols, namely, those including coin tossing and intrinsic cryptographic primitives (such as zero-knowledge proof systems, oblivious transfer, secure computation, and bit commitment).

Moreover, it is well known that there are protocols that can be proved secure for the $\mathcal{DY}$ model, but are insecure when considering specific cryptosystems. For example, a "correct" implementation of the Needham-Schroder-Lowe protocol can be attacked if El-Gamal is used as the underlying cryptosystem [15]. However, problems with the real implementation of the protocols are not addressed by the current generation of security protocol analysis tools, and even when results that relate symbolic and complexity models [1, 2] are exploited, the equivalence is up to a negligible function and so attacks might be found when we are in the "real world".

The main goal of our work is to further close the gap between formal analysis and concrete implementations of security protocols by introducing a quantitative extension of the usual Dolev-Yao model, which we call $\mathcal{DY}^+$. This allows us to consider protocol attacks that are possible when the intruder has a reasonable amount of computational power, in particular when he is able to guess encryption keys.

We formalize such an intruder model by extending the $\mathcal{DY}$ model with additional structure and intruder capabilities. In particular, while in the $\mathcal{DY}$ model encryption is a "black box" operation that can only be undone with the right decryption key, in the $\mathcal{DY}^+$ model we introduce new intruder-derivation rules, which depend on the encryption scheme that is being used, and thus associate more structure to each encryption, such as the corresponding key space.

Moreover, when using the $\mathcal{DY}$ model, it is customary to formalize interleaved executions of a protocol as an infinite-state transition system, which the tool can then search for states that represent protocol attacks (e.g. when the intruder gets hold of some data that was intended to be a secret between two honest agents). This transition system defines a computation tree in the standard way, where the root is the initial system state and children represent the ways that a state can evolve in one transition. In the $\mathcal{DY}^+$ model, we extend this search tree so that each possible transition is weighed with a probability; more specifically, 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 tree will be weighed with the probability of that transition, we can compute the probabilities associated with the branches of the tree, and in the end we will be able to tell with which probability each attack is possible.

Another contribution of our work is to show that the computational complexity of protocol insecurity with finite numbers of protocol sessions is not augmented by our extensions. As shown, for instance, in [11], searching for an attack in the classical $\mathcal{DY}$ intruder model is an NP-complete problem when considering finite numbers of

protocol sessions, and we have shown that if we consider the extended probabilistic intruder model $\mathcal{DY}^+$, the problem remains NP-complete.

2 Extending the Intruder Model

Our approach is general and technology-independent, and can thus be effectively incorporated in different techniques and tools for security protocol analysis. For concreteness, we consider the constraint-based, *on-the-fly-model-checker OFMC* [3, 4, 14][1] and we will now briefly illustrate how we have extended the key ideas, definitions and results that underlie OFMC to deal with the augmented intruder model.

The classical Dolev-Yao model assumes that cryptography is perfect and that the intruder can only decrypt an encrypted message if he knows the corresponding decryption key. Since one of the ultimate goals of our work is to augment the intruder model in such a way to allow the intruder to perform attacks on cryptography (e.g. representing that he can break an encryption with a certain probability), we can no longer ignore the underlying cryptographic schemes used in the protocol specification.

We allow the attacker to "retrieve" keys to decrypt encrypted messages, but only with some probability of success. The intruder can then generate and analyze protocol messages using this new piece of information, which may possibly result in a significant increase of the intruder knowledge and thus open new ways for attacks. It is therefore important to accurately measure the probability of correctly retrieving the encryption key, either by simple guessing or resorting to cryptanalysis techniques.

In general, with this new intruder model we expect to detect flaws in protocols caused by badly chosen cryptographic systems in important positions of the protocol flow, such as a 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. We decided to empower our intruder 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 efforts of the intruder, 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 consider the context-free grammar given in [4] for formalizing protocol descriptions, and extend it to express the probability of a transition when a key is guessed by the intruder given his current knowledge. We introduce a function $p : \mathcal{L}(Msg) \times \mathcal{P}(\mathcal{L}(PosFact)) \to \mathbb{R}$ that 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 $\mathsf{i\_knows}(Msg)$. Msg is the set of possible messages (including message components such as agent names, nonces, keys, etc.) and, as discussed in more detail in [4], we use positive facts to represent either the local state of an honest agent ($\mathsf{state}(Msg)$), a message in transit through the network (i.e. one sent but not yet received, $\mathsf{msg}(Msg)$), a message known by the intruder ($\mathsf{i\_knows}(Msg)$), or a secret message ($\mathsf{secret}(Msg, Msg)$). Here, we additionally consider positive facts $\mathsf{plabel}(\mathbb{R})$ to capture key-guessing as described above.[2]

The way the intruder increases his knowledge needs also to be extended with respect to [4] to reflect the new model (and the new kinds of possible attacks). Essentially, the way the intruder reasons will only be affected by the new key-guessing ability. We thus maintain the set of intruder-deduction rules of [4], which represent that the intruder can use pairing, function application, symmetric ($\{|\cdot|\}$) and asymmetric ($\{\cdot\}$) encryption to compose and decompose messages, and extend it with two guessing rules for encryption keys, which represent the possibility of the intruder breaking a symmetric or asymmetric encryption and gaining knowledge of the encrypted message together with the encryption key.

Formally, for a set M of messages, we define $\mathcal{DY}^+(M)$ as the smallest set closed under the following generation (G) and analysis (A) rules together with the two rules guessS and guessPK for the intruder guessing symmetric

[1] OFMC has been developed in the context of the AVISPA project. The AVISPA Tool is a fully automated protocol analysis environment that comprises of OFMC and three other tools (called CL-AtSe, SATMC, and TA4SP) and that has been successfully applied for the push-button analysis of a large number of industrial-scale security protocols.

[2] Note that 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. Note also that the approach we propose here is different from, and is in fact complementary to, the symbolic, deductive approach to off-line guessing proposed in [8].

and public keys:

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

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

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

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

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

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

A protocol is then represented by a triple (I, R, AR), where I is the initial state, R is the set of transition rules and AR is a rule for identifying an attack state. We build the infinite-state transition system modeling interleaved executions of a protocol by starting from an initial state I, which contains the initial intruder knowledge and the knowledge of the honest protocol agents, and computing each successor state by applying the transition rules in R. An attack-rule of a protocol describes the condition(s) under which an attack takes place, and thus describes the attack-states of the protocol execution.

To incorporate the changes in the intruder model, every state transition must reflect the fact that the intruder may have used a guessing rule. Thus, we need to distinguish the deterministic transitions from the probabilistic ones. To carry out this distinction, we label each transition with the value we obtain from the function p, where p returns 1 if no guessing was needed. When a state S is found that triggers the attack-rule AR, we determine the attack probability by traversing backwards the attack trace (from S to I) and multiplying all the probability labels. Note that by doing so, we only "introduce" new attacks with respect to the analysis in the $\mathcal{DY}$ model; the "old" ones will be detected and will be labeled with probability 1.

The rules in the set R describe state transitions. The left-hand side *lhs* of a rule $r = lhs \Rightarrow rhs$ consists of a set of positive facts P, a set of negative facts N, and a condition *Cond*, where $vars(P) \supseteq vars(N) \cup vars(Cond)$. A rule is applicable to a state if (i) the positive facts are contained in the state for some substitution σ of the rule's variables, (ii) the negative facts under σ are not contained, and (iii) the condition *Cond* is satisfied under σ. The right-hand side *rhs* of a rule $lhs \Rightarrow rhs$ is just a set of positive facts, where we require that $vars(lhs) \supseteq vars(rhs)$. The successors of a state S are then the states generated by replacing in S the facts that match the positive facts of the *lhs* of some applicable rule with the *rhs* of that rule.

Formally, we consider rules $r = lhs \Rightarrow rhs$ of the form

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

where N_1 is a set of negative facts that do not contain i_knows or msg facts, P_1 and P_2 are sets of positive facts that do not contain state or msg facts, and *Cond* is a condition, i.e. a conjunction of inequalities of messages. Moreover, we require that if $\mathsf{i\_knows}(m) \in P_1$ then $\mathsf{i\_knows}(m) \in P_2$; this ensures that the intruder knowledge is *monotonic*, i.e. that the intruder never forgets messages during transitions.

More specifically, every such rule describes a transition of an honest agent, since a state fact appears in both the *lhs* and the *rhs* of the rule. Also, in both sides we have a msg fact representing the incoming message that the agent expects to receive in order to make the transition (in the *lhs*) and the agent's answer message (in the *rhs*). The rule corresponding to the initial (respectively, final) protocol step contains no incoming (respectively, outgoing) message. However, the above rule form is not a restriction, as one may always insert a dummy message

that can be generated by the intruder. In fact, as we argue in [4], rules of the above form are adequate to describe a very large class of industrial-scale security protocols.

Further, let $\overline{P_1}$ be obtained from P_1 by removing all i_knows facts

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

We define the applicability of 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:

$$\begin{aligned} \mathit{applicable}_{\mathit{lhs}}(S) = \{\langle \sigma, \xi \rangle \mid \\ & \mathrm{ground}(\sigma) \wedge \mathrm{dom}(\sigma) = \mathrm{vars}(m_1) \cup \mathrm{vars}(m_2) \cup \mathrm{vars}(P_1) \\ & \wedge \{m_1\sigma\} \cup \{m\sigma \mid \mathsf{i\_knows}(m) \in P_1\} \subseteq \mathcal{DY}^+(\{m \mid \mathsf{i\_knows}(m) \in S\}) \\ & \wedge \xi = \Pi_{t \in \text{co-dom}(\sigma)} p(t, \mathcal{DY}^+(\{m \mid \mathsf{i\_knows}(m) \in S\})) \\ & \wedge \mathsf{state}(m_2\sigma) \in S \wedge \overline{P_1}\sigma \subseteq \\ & \wedge (\forall f.\ \mathsf{not}(f) \in N_1 \Longrightarrow f\sigma \notin S) \wedge \sigma \vdash \mathit{Cond}\}. \end{aligned}$$

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 step function

$$\begin{aligned} \mathit{step}_{\mathit{lhs} \Rightarrow \mathit{rhs}}(S) = \{S' \mid \exists \sigma.\, \exists \xi. \\ & \langle \sigma, \xi \rangle \in \mathit{applicable}_{\mathit{lhs}}(S) \\ & \wedge S' = (S \setminus (\mathsf{state}(m_2\sigma) \cup \overline{P_1}\sigma)) \cup \mathsf{i\_knows}(m_4\sigma) \cup P_2\sigma \cup \mathsf{plabel}(\xi)\}. \end{aligned}$$

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 and then to its successors. In this way, 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.

Formally, the set of reachable states of the protocol (I, R, AR) is the set

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

Given an attack-rule AR and a state S, the attack predicate $\mathit{isAttack}_{AR}(S)$ is true if and only if the rule AR can be applied to the state S, i.e. $\mathit{applicable}_{AR}(S) \neq \emptyset$. Moreover, given an attack-rule AR, if $\mathit{isAttack}_{AR}(S)$ is true, then the probability $\mathit{pAttack}_{AR}(S)$ of such an attack is given by

$$\mathit{pAttack}_{AR}(S) = \prod_{x \in \{t \mid \mathsf{plabel}(t) \in S\}} x\,.$$

Definition 1 (Attack Problem for k sessions in the $\mathcal{DY}^+$ model). Given a protocol (I, R, AR) in the $\mathcal{DY}^+$ model, determine whether this protocol has an attack, with probability greater than p, as defined by AR when k sessions of the protocol are run.

The extension we propose does not change the model-checking procedure, apart from the probability labels introduced in the transition-system states. Moreover, it is not difficult to show that the complexity class for the Attack Problem for k sessions remains in NP for the extended Dolev-Yao intruder model $\mathcal{DY}^+$.

Theorem 1. The Attack Problem for k sessions in the $\mathcal{DY}^+$ intruder model is NP-complete.

The extension we propose requires a clear commitment to the cryptosystem being used in the specified protocol, that is, the size of the keys and the key space. This information is important because it is required in the calculation of the transition probabilities and the attack probability. We can thus extend OFMC, as we are currently doing, by extending the underlying model-checking procedure with a calculation of probabilities. This will allow us to employ the tool to find out if the attack probability is below a given ϵ, and reduce the search space by abandoning the state exploration when the product of the probability labels of a certain state is less than that given ϵ.

References

[1] M. Abadi and P. Rogaway. Reconciling two views of cryptography: The computational soundness of formal encryption. In *Proceedings of 1st IFIP International Conference on Theoretical Computer Science*, LNCS 1872, pages 3–22, 2000.

[2] P. Adão, G. Bana, and A. Scedrov. Computational and information-theoretic soundness and completeness of formal encryption. In *Proceedings of CSFW'05*, pages 170–184, 2005.

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

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

[5] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. CSFW'01*. IEEE Computer Society Press, 2001.

[6] C. Cremers. Scyther documentation. `http://www.win.tue.nl/~ccremers/scyther`.

[7] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[8] P. Hankes Drielsma, S. Mödersheim, and L. Viganò. A formalization of off-line guessing for security protocol analysis. In *Proceedings of LPAR'04*, LNAI 3452, pages 363–379. Springer, 2005.

[9] C. Meadows. The NRL Protocol Analyzer: An Overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[10] L. C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6(1):85–128, 1998.

[11] M. Rusinowitch and M. Turuani. Protocol Insecurity with Finite Number of Sessions is NP-complete. In *Proceedings of CSFW'01*. IEEE Computer Society Press, 2001. Available at `http://www.avispa-project.org`.

[12] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *Modelling and Analysis of Security Protocols*. Addison Wesley, 2000.

[13] D. Song. Athena: A new efficient automatic checker for security protocol analysis. In *Proc. CSFW'99*. IEEE Computer Society Press, 1999.

[14] L. Viganò. Automated Security Protocol Analysis with the AVISPA Tool. In *Proceedings of the XXI Mathematical Foundations of Programming Semantics (MFPS'05)*. ENTCS, to appear.

[15] B. Warinschi. A computational analysis of the Needham-Schroeder-(Lowe) protocol. In *Proceedings of CSFW'03*, pages 248–262, 2003.