Formal Verification of Cryptographic Protocols

Steve Kremer

Laboratoire Spécification et Vérification
ENS Cachan & CNRS & INRIA Futurs projet SECSI, France
kremer@lsv.ens-cachan.fr

1 Introduction

Security protocols or *cryptographic protocols* are small distributed programs that ensure security properties in a hostile environment. Typical examples of properties that need to be ensured are *secrecy* and *authentication*. For instance when we consider a home banking application, we want to ensure that the information is indeed sent to our bank—this is an authentication problem—and that the information remains confidential—this is a secrecy property. Sometimes, we may also require more sophisticated properties. In our home-banking example, we may think about *non-repudiation*: it is not possible to deny, or repudiate later that some transaction has been carried out, i.e., there exists some data, typically a digital signature, that can be invoked as a proof that a given transaction has been requested. Another interesting security property one can think of is anonymity. In this tutorial, for the sake of simplicity, we will focus on secrecy.

Security protocols rely on cryptographic primitives, such as symmetric and asymmetric encryption, digital signatures and one-way hash functions. We here assume a basic familiarity with these primitives (without however requiring a deep understanding of the algorithms which realize them). The interested reader can find more details about such algorithms in [19]. History has demonstrated that security protocols are tremendously error-prone, even for very short protocols. The most famous example is probably the Needham-Schroeder public-key protocol [23], a 3-line protocol, for which Lowe [17] found an attack 17 years after its publication.

The seminal paper of Dolev and Yao [15] has initiated the use of formal methods to validate security protocols. The two main ideas of this model are: (*i*) the adversary has complete control of the network, i.e., he can remove messages and insert any message he can construct; (*ii*) the cryptography is perfect, typically it is impossible to undo an encryption or learn anything from the plaintext, unless the adversary knows the decryption key. The data sent in the protocol and manipulated by the intruder is formalized using an *abstract algebra*. Following this approach, there have been a large variety of formalisms to reason about the correctness of security protocols. In this tutorial we do not aim to review all of these methods, but we specifically focus on some recent, fully automated techniques.

2 Abstract term algebras

As explained above, in the Dolev-Yao approach, we model the protocol messages and cryptographic primitives using an abstract term algebra. A term algebra is defined by a signature $\mathcal{F}$ and a set of variables $\mathcal{X}$. The signature $\mathcal{F}$ is a set of function symbols, given each with an arity (we suppose that all function applications respect this arity). Function symbols of arity 0 model constants. The set of ground terms (containing no variables) is denoted $\mathcal{T}(\mathcal{F})$ and the set of all terms is denoted $\mathcal{T}(\mathcal{F}, \mathcal{X})$.

An example of a useful signature in security protocols consists in the two symbols $\mathsf{enc}/2$ and $\mathsf{pair}/2$. The first models encryption of a message with a key while the second models pairing. Classically there have been two semantic models that have been considered. One of this model considers a *free term algebra*, i.e., each distinct term has exactly one representation. This free term algebra would be equipped with deduction rules

$$\frac{\mathsf{enc}(x,y) \quad y}{x} \qquad \frac{\mathsf{pair}(x,y)}{x} \qquad \frac{\mathsf{pair}(x,y)}{y}$$

stating that an adversary can decrypt an encryption if he knows the encryption key (here y) and access the components of a pair. One can also give the semantics of a term algebra (not free this time) by an *equational theory*. In that case we have to model *explicit destructors*. In our example theory we extend the signature with the function symbols $\mathsf{dec}/2$, $\mathsf{fst}/1$ and $\mathsf{snd}/1$ and define the equational theory E

$$\mathsf{dec}(\mathsf{enc}(x,y),y) = x \qquad \mathsf{fst}(\mathsf{pair}(x,y)) = x \qquad \mathsf{snd}(\mathsf{pair}(x,y)) = y.$$

A term may now have different representations, e.g., $\mathsf{dec}(\mathsf{enc}(\mathsf{fst}(\mathsf{pair}(x,y)),z),z)$ and x are equal modulo E (denoted $=_E$). In [20, 18], it has been shown that some attacks are only present when explicit destructors are used (this is because the adversary can construct terms such as $\mathsf{dec}(x,y)$, where x is not an encryption). We therefore adopt this model in the remaining of this tutorial.

3 Difficulties when verifying cryptographic protocols

Even, when considering a Dolev-Yao model with perfect cryptography, automatic verification is not always feasible. The following difficulties arise when automating the verification of security protocols:

- the intruder can build an infinite number of messages (for instance iterating encryption);
- there may be an unbounded number of parallel sessions;
- the equational theories manipulated by the intruder may differ and add complications.

It is by now well known that verification of security protocols in general is undecidable, see for instance [16], even when we consider a simple equational

theory only modeling encryption and pairing. There exist nevertheless several approaches to deal with the problem. These approaches either consider weakened adversaries or propose algorithms that do not guarantee termination and/or completeness, i.e., they may fail to prove the correctness of a valid protocol.

4 Passive adversaries

An interesting special case is to consider a *passive* adversary or *eavesdropper*. A passive adversary does not interact with the protocol, but merely observes the exchanged messages. This eliminates the two first difficulties stated above. The question we propose to investigate is the following: given a set of ground terms $S = \{M_1, \ldots, M_n\}$, is the adversary able to deduce a secret term s, generally denoted $S \vdash_E s$. Whether this problem is decidable and the complexity of this problem rely on the equational theory E. This problem is decidable for many useful theories and can even be solved efficiently in many cases, e.g., [11, 13, 1].

Sometimes, we are not interested to merely know whether a value is deducible, but rather whether an adversary can distinguish two tuples of terms $S_1 = \{M_1, \ldots, M_n\}$ and $S_2 = \{N_1, \ldots, N_n\}$. If the adversary is unable to produce a *test* that distinguishes these two tuples, we say that they are statically equivalent [3], written $S_1 \approx_E S_2$. Again, decidability and complexity depend on the equational theory. Decidability is known for an interesting class of theories [1, 2] (for many of these theories the algorithms are even polynomial). How an equational theory can be implemented in a sound way with respect to $\approx_E$ and $\vdash_E$ is discussed in [5].

5 Active adversary with finite number of sessions

Restricting the number of sessions the adversary can initiate yields decidability (for many theories at least) even without limiting the messages the intruder can construct. Many recent works are based on *constraint solving*. A protocol will be presented by a number of *roles*. A role itself is a sequence of send and receive actions with possibly some tests on the received messages. Typically a protocol role R of size m_R will be a sequence of the form

$$\mathsf{receive}(x_i) \qquad (M_1^i =_E N_1^i, \ldots, M_{\ell_i}^i =_E N_{\ell_i}^i) \qquad \mathsf{send}(T_i)$$

where M_j^i, N_j^i and T_i are terms whose variables are included in $\{x_j \mid j \leq i\}$ for $1 \leq i \leq m_R$. Any consistent (respecting the local order of each role) interleaving of size n of protocol roles generates constraints of the form $\{S_{i-1} \vdash_E x_i, M_1^i =_E N_1^i, \ldots, M_{\ell_i}^i =_E N_{\ell_i}^i\}$ $(1 \leq i \leq n)$ where S_0 is the initial knowledge of the attacker and $S_i = S_{i-1} \cup \{T_{i-1}\}$. To express the secrecy of a term s we add a final constraint $S_{n+1} \vdash_E s$. The protocol is insecure, if the attacker can find an interleaving for which he can solve the constraint system, i.e., find a substitution for the x_is. In [24], Turuani and Rusinowitch have shown that the problem is NP-complete for a classical theory of encryption and pairs. Such a constraint

solving algorithm has been implemented in [21] and optimized in [12]. There have also been many procedures for more specialized intruder theories [8–10, 22, 14]. In [13, 4], the authors give decision procedures for a class of equational theories in a framework with explicit destructors.

6 Active adversary with unbounded number of sessions

A popular formalism for protocol verification with an unbounded number of sessions are first-order Horn clauses. We will concentrate here on the work of Blanchet [7] as his work has been implemented in the ProVerif tool [6], one of the state-of-the-art tools for verifying protocols. The basic idea is very simple: we define a special predicate $I(x)$, meaning that the intruder knows x. The capacities of the intruder can then be encoded using Horn clauses. For example the theory of pairs yields the following Horn clauses:

$$\frac{I(x) \quad I(y)}{I(\mathsf{pair}(x,y))} \qquad \frac{I(x)}{I(\mathsf{fst}(x))} \qquad \frac{I(x)}{I(\mathsf{snd}(x))} \qquad \frac{I(\mathsf{fst}(\mathsf{pair}(x,y)))}{I(x)} \qquad \frac{I(\mathsf{snd}(\mathsf{pair}(x,y)))}{I(y)}$$

In a similar way, the protocol rules can be encoded: whenever the intruder "knows" a term matching a send action, he can learn the term corresponding to the receive action. This way of modeling, however, introduces some imprecision as we do not account for the order of the protocol messages and only consider a finite set of nonces (random values) which depend on the parameters instantiating the protocol. However, all these abstractions are safe with respect to proving a protocol correct, as they give additional power to the intruder.

References

1. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. In *Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, volume 3142 of *LNCS*, pages 46–58. Springer, 2004.
2. M. Abadi and V. Cortier. Deciding knowledge in security protocols under (many more) equational theories. In *Proc. 18th Computer Security Foundations Workshop (CSFW'05)*, pages 62–76. IEEE, 2005.
3. M. Abadi and C. Fournet. Mobile values, new names, and secure communications. In *Proc. 28th Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM, 2001.
4. M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proc. 12th Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM, 2005.
5. M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *LNCS*, pages 652–663. Springer, 2005.
6. B. Blanchet. Personal web page. http://www.di.ens.fr/ blanchet.
7. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE, 2001.

8. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents. In *Proc. 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST–TCS'03)*, volume 2914 of *LNCS*, pages 124–135, 2003.
9. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An np decision procedure for protocol insecurity with xor. In *Proc. of 18th Symposium on Logic in Computer Science (LICS '03)*. IEEE, 2003.
10. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. 18th Symposium on Logic in Computer Science (LICS '03)*, pages 271–280. IEEE, 2003.
11. H. Comon-Lundh and R. Treinen. Easy intruder deductions. In *Verification: Theory and Practice, Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday*, volume 2772 of *Lecture Notes in Computer Science*, pages 225–242. Springer, 2003. Invited paper.
12. R. Corin and S. Etalle. An improved constraint-based system for the verification of security protocols. In *Proc. 9th International Static Analysis Symposium (SAS'02)*, volume 2477 of *LNCS*, pages 326–341. Springer, 2002.
13. S. Delaune and F. Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In *Proc. 11th Conference on Computer and Communications Security (CCS'04)*, pages 278–287. ACM, 2004.
14. S. Delaune, P. Lafourcade, D. Lugiez, and R. Treinen. Symbolic protocol analysis in presence of a homomorphism operator and *Exclusive Or*. In *Proc. 33rd International Colloquium on Automata, Languages and Programming (ICALP'06)*, LNCS. Springer, 2006. To appear.
15. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(12):198–208, 1983.
16. N. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.
17. G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.
18. C. Meadows and C. Lynch. On the relative soundness of the free algebra model for public key encryption. In *Proc. of the 4th Workshop on Issues in the Theory of Security (WITS'04)*, 2004.
19. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. Series on Discrete Mathematics and its Applications. CRC Press, 1997.
20. J. K. Millen. On the freedom of decryption. *Information Processing Letters*, 86(6):329–333, 2003.
21. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th Conference on Computer and Communications Security (CCS'01)*, pages 166–175. ACM, 2001.
22. J. K. Millen and V. Shmatikov. Symbolic protocol analysis with an abelian group operator or diffie-hellman exponentiation. *Journal of Computer Security*, 13(3):515–564, 2005.
23. R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.
24. M. Turuani and M. Rusinowitch. Protocol insecurity with finite number of sessions is NP-complete. In S. Schneider, editor, *14th Computer Security Foundations Workshop*, pages 174–187. IEEE, 2001.