Verification of Security Protocols

Véronique Cortier

LORIA, CNRS, Nancy, France*

1 Introduction

Security protocols are short programs aiming at securing communications over a network. They are widely used in our everyday life. They may achieve various goals depending on the application: confidentiality, authenticity, privacy, anonymity, fairness, etc. Their verification using symbolic models has shown its interest for detecting attacks and proving security properties. A famous example is the Needham-Schroeder protocol [23] on which G. Lowe discovered a flaw 17 years after its publication [20]. Secrecy preservation has been proved to be co-NP-complete for a bounded number of sessions [24], and decidable for an unbounded number of sessions under some additional restrictions (*e.g.* [3, 12, 13, 25]). Many tools have also been developed to automatically verify cryptographic protocols like [21, 8].

In this tutorial, we first overview several techniques used for symbolically verifying security protocols that have led to the design of many efficient and useful tools. However, the guarantees that symbolic approaches offer have been quite unclear compared to the computational approach that considers issues of complexity and probability. This later approach captures a strong notion of security, guaranteed against all probabilistic polynomial-time attacks. In a second part of the tutorial, we present recent results that aim at obtaining the best of both worlds: fully automated proofs and strong, clear security guarantees. The approach consists in proving that symbolic models are *sound* with respect to computational ones, that is, that any potential attack is indeed captured at the symbolic level.

2 Symbolic Approach

In symbolic models, the implementation details of the primitives are abstracted away, and the execution is modeled only symbolically. The central characteristics of the symbolic approach is that messages are modeled using a term algebra T^f. Typically, the messages exchanged by parties are symbolic terms constructed from symbols for nonces n, identities a, by applying abstract operations representing encryption $\mathsf{enc}(m, k)$, pairing $\langle m_1, m_2 \rangle$, signature $\mathsf{sign}(m, k)$, etc. Specifically, we consider the *signature* $\mathcal{F} = \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}, \langle\,\rangle, \mathsf{pub}, \mathsf{priv}\}$ together with

* This work has been partially supported by the ARA project AVOTÉ and the ARA SSIA FormaCrypt.

arities of the form $\mathsf{ar}(f) = 2$ for the four first symbols and $\mathsf{ar}(f) = 1$ for the two last ones. The symbol $\langle\,\rangle$ represents the pairing function. The terms $\mathsf{enc}(m, k)$ and $\mathsf{enca}(m, k)$ represent respectively the message m encrypted with the symmetric (resp. asymmetric) key k. The term $\mathsf{sign}(m, k)$ represents the message m signed by the key k. The terms $\mathsf{pub}(a)$ and $\mathsf{priv}(a)$ represent respectively the public and private keys of an agent a. We fix an infinite set of *variables* $\mathcal{X} = \{x, y \ldots\}$. Terms with variables are used to specify the intended behavior of the protocol. The set T^f of *terms* is defined inductively by

$t ::=$		term
	x	variable x
	a	name a
	$f(a)$	application of symbol $f \in \{\mathsf{pub}, \mathsf{priv}\}$ on a name
	$f(t_1, t_2)$	application of symbol $f \in \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}, \langle\,\rangle\}$

The set of variables occurring in a term t is denoted by $\mathcal{V}(t)$. The set of *subterms* of a term t is denoted by $\mathsf{st}(t)$.

2.1 Intruder Deduction

Public network are insecure. We assume that a potential attacker, also called intruder, can not only listen to the network but also intercept, block and send new messages. The way an intruder can learn and build new messages is typically modeled through a deduction system. We give an example of such a deduction system in Figure 1. It corresponds to the usual Dolev-Yao rules. The first line describes the *composition* rules, the two last lines the *decomposition* rules. Intuitively, these deduction rules say that an intruder can compose messages by pairing, encrypting and signing messages provided he has the corresponding keys and conversely, he can decompose messages by projecting or decrypting provided he has the decryption keys. For signatures, the intruder is also able to *verify* whether a signature $\mathsf{sign}(m, k)$ and a message m match (provided he has the verification key), but this does not give him any new message. That is why this capability is not represented in the deduction system. We also consider a rule

$$\frac{S \vdash \mathsf{sign}(x, \mathsf{priv}(y))}{S \vdash x}$$

that expresses that an intruder can retrieve the whole message from his signature. This property may or may not hold depending on the signature scheme.

A term u is *deducible* from a set of terms S, denoted by $S \vdash u$ if there exists a *proof i.e.* a tree such that the root is $S \vdash u$, the leaves are of the form $S \vdash v$ with $v \in S$ (*axiom* rule) and every intermediate node is an instance of one of the rules of the deduction system.

$$\frac{S \vdash x \quad S \vdash y}{S \vdash \langle x, y \rangle} \qquad \frac{S \vdash x \quad S \vdash y}{S \vdash \mathsf{enc}(x, y)} \qquad \frac{S \vdash x \quad S \vdash y}{S \vdash \mathsf{enca}(x, y)} \qquad \frac{S \vdash x \quad S \vdash y}{S \vdash \mathsf{sign}(x, y)}$$

$$\frac{S \vdash \langle x, y \rangle}{S \vdash x} \qquad \frac{S \vdash \langle x, y \rangle}{S \vdash y} \qquad \frac{S \vdash \mathsf{enc}(x, y) \quad S \vdash y}{S \vdash x}$$

$$\frac{S \vdash \mathsf{enca}(x, \mathsf{priv}(y)) \quad S \vdash \mathsf{pub}(y)}{S \vdash x} \qquad \frac{S \vdash \mathsf{sign}(x, \mathsf{priv}(y))}{S \vdash x}$$

Fig. 1. Intruder Deduction System.

Example 1. The term $\langle k_1, k_2 \rangle$ is deducible from the set $S_1 = \{\mathsf{enc}(k_1, k_2), k_2\}$. Indeed, a proof corresponding to that fact that $S_1 \vdash \langle k_1, k_2 \rangle$ is:

$$\dfrac{\dfrac{S_1 \vdash \mathsf{enc}(k_1, k_2) \quad S_1 \vdash k_2}{S_1 \vdash k_1} \qquad S_1 \vdash k_2}{S_1 \vdash \langle k_1, k_2 \rangle}$$

2.2 Protocols - Bounded Number of Sessions

Consider the famous Needham-Schroeder asymmetric key authentication protocol [23] designed for mutual authentication.

$$\begin{array}{ll} A \to B: & \mathsf{enca}(\langle N_A, A \rangle, \mathsf{pub}(B)) \\ B \to A: & \mathsf{enca}(\langle N_A, N_B \rangle, \mathsf{pub}(A)) \\ A \to B: & \mathsf{enca}(N_B, \mathsf{pub}(B)) \end{array}$$

The agent A sends to B his name and a fresh nonce (a randomly generated value) encrypted with the public key of B. The agent B answers by copying A's nonce and adds a fresh nonce N_B, encrypted by A's public key. The agent A acknowledges by forwarding B's nonce encrypted by B's public key. This protocol defines two *roles* that specify the behavior of the initiator agent A and the behavior of the responder agent B. Formally, the executions of a protocol are specified using terms with variables. We consider for simplicity a scenario where A starts a session with a corrupted agent I (whose private key is known to the intruder) and B is willing to answer to A. The initial knowledge of the intruder is

$$T_0 = \{a, b, i, \mathsf{pub}(a), \mathsf{pub}(b), \mathsf{pub}(i), \mathsf{priv}(i)\}.$$

The set $T_1 \stackrel{\mathsf{def}}{=} T_0 \cup \{\mathsf{enca}(\langle n_a, a \rangle, \mathsf{pub}(i))\}$ represents the messages known to the intruder once A has contacted the corrupted agent I. Then if a message of the form $\mathsf{enca}(\langle x, a \rangle, \mathsf{pub}(b))$ can be sent on the network, then B would answer to this message by $\mathsf{enca}(\langle x, n_b \rangle, \mathsf{pub}(a))$, in which case the knowledge of the intruder will turn to be

$$T_2 \stackrel{\mathsf{def}}{=} T_1 \cup \{\mathsf{enca}(\langle x, n_b \rangle, \mathsf{pub}(a))\}.$$

Subsequently, if a message of the form $\mathsf{enca}(\langle n_a, y\rangle, \mathsf{pub}(a))$ can be sent on the network, then A would answer by $\mathsf{enca}(y, \mathsf{pub}(i))$ since A believes she is talking to I and the knowledge of the intruder would be represented by

$$T_3 \stackrel{\mathsf{def}}{=} T_2 \cup \{\mathsf{enca}(y, \mathsf{pub}(i))\}.$$

The run corresponds to an attack if (for example), the intruder is able to learn the nonce n_b.

This execution can be formally modeled by a *constraint system*.

Definition 1. *A* constraint system C *is a finite set of expressions* $T_i \Vdash u_i$*, where* T_i *is a non empty set of terms and* u_i *is a term,* $1 \leq i \leq n$*, such that:*

- $T_i \subseteq T_{i+1}$*, for all* $1 \leq i \leq n-1$*;*
- *if* $x \in \mathcal{V}(T_i)$ *then* $\exists j < i$ *such that* $T_j = \min\{T \mid (T \Vdash u) \in C, x \in \mathcal{V}(u)\}$ *(for the inclusion relation) and* $T_j \subsetneq T_i$*.*

The first condition says that the intruder knowledge only increases. The second condition ensures that variables are always first introduced on the right-hand side of a constraint. This corresponds to the fact that the output of a protocol depends deterministically on its entry.

A solution of a constraint system C is a substitution θ such that $\forall (T \Vdash u) \in C$, $T\theta \vdash u\theta$ holds.

Continuing our example, the set constraint corresponding to our scenario is:

$$T_1 \stackrel{\mathsf{def}}{=} T_0 \cup \{\mathsf{enca}(\langle n_a, a\rangle, \mathsf{pub}(i))\} \Vdash \mathsf{enca}(\langle x, a\rangle, \mathsf{pub}(b)) \quad (1)$$

$$T_2 \stackrel{\mathsf{def}}{=} T_1 \cup \{\mathsf{enca}(\langle x, n_b\rangle, \mathsf{pub}(a))\} \Vdash \mathsf{enca}(\langle n_a, y\rangle, \mathsf{pub}(a)) \quad (2)$$

$$T_3 \stackrel{\mathsf{def}}{=} T_2 \cup \{\mathsf{enca}(y, \mathsf{pub}(i))\} \Vdash n_b \quad (3)$$

Checking properties like confidentiality is thus reduced to finding a solution to set constraints. In our running example, a solution to the above set constraints corresponds to an attack on the confidentiality of the nonce n_b.

A way to solve set constraints is to transform them step by step into simpler ones [22]. A variant of the transformation rules, proposed by Hubert Comon-Lundh, are displayed in Figure 2. Using transformation rules, it can be shown that solving set constraints is an NP-complete problem [24, 16]. It should be noticed that solving set constraints corresponding to checking security for one particular scenario. Once the number of sessions is fixed, there is finitely many (polynomially guessable) number of scenarios. This yields NP-completeness of secrecy for protocols for a bounded number of sessions.

2.3 Unbounded Number of Sessions

In the general case - when the number of sessions is not fixed - even a simple property such as secrecy is undecidable [18]. A classical restriction is to consider a bounded number of nonces, meaning that the same nonce may be reused in

$$
\begin{array}{lllll}
R_1 & C \wedge T \Vdash u & \rightsquigarrow & C & \text{if } T \cup \{x \mid (T' \Vdash x) \in C, T' \subsetneq T\} \vdash u \\
R_2 & C \wedge T \Vdash u & \rightsquigarrow_\sigma & C\sigma \wedge T\sigma \Vdash u\sigma & \text{if } \sigma = \mathsf{mgu}(t,u),\ t \in \mathsf{st}(T), \\
 & & & & \quad t \neq u,\ t,u \text{ not variables} \\
R_3 & C \wedge T \Vdash u & \rightsquigarrow_\sigma & C\sigma \wedge T\sigma \Vdash u\sigma & \text{if } \sigma = \mathsf{mgu}(t_1,t_2),\ t_1,t_2 \in \mathsf{st}(T), \\
 & & & & \quad t_1 \neq t_2,\ t_1,t_2 \text{ not variables} \\
R_4 & C \wedge T \Vdash u & \rightsquigarrow & \bot & \text{if } \mathcal{V}(T,u) = \emptyset \text{ and } T \not\vdash u \\
R_f & C \wedge T \Vdash f(u,v) & \rightsquigarrow & C \wedge T \Vdash u \wedge T \Vdash v & \text{for } f \in \{\langle\,\rangle, \mathsf{enc}, \mathsf{enca}, \mathsf{sign}\}
\end{array}
$$

Fig. 2. Simplification Rules.

different sessions, while this does not hold in reality. Such an abstraction may lead to false attack but allows to *prove* security properties [8, 9] for an unbounded number of sessions. Checking properties like secrecy remains undecidable [18] but it enables to represent protocol execution using Horn clauses.

Typically, we consider a predicate I that represents the intruder knowledge. For example, the initial knowledge of the intruder for the Needham-Schroeder protocol can be modeled by the set

$$\mathcal{C}_{I_0} = \{I(a), I(b), I(i), I(\mathsf{pub}(a)), I(\mathsf{pub}(b)), I(\mathsf{pub}(i)), I(\mathsf{priv}(i))\}.$$

Abstracting nonces by constants, an unbounded execution of the Needham-Schroeder protocol can be represented by the following set $\mathcal{C}_{\mathsf{NS}}$ of clauses:

$$
\begin{array}{r}
\Rightarrow I(\mathsf{enca}(\langle n_a, a\rangle, \mathsf{pub}(i))) \\
I(\mathsf{enca}(\langle x, a\rangle, \mathsf{pub}(b))) \Rightarrow I(\mathsf{enca}(\langle x, n_b\rangle, \mathsf{pub}(a))) \\
I(\mathsf{enca}(\langle n_a, y\rangle, \mathsf{pub}(a))) \Rightarrow I(\mathsf{enca}(y, \mathsf{pub}(i)))
\end{array}
$$

For simplicity, we have only described the clauses corresponding to the case where A starts sessions with a corrupted agent I and B is willing to answer to A. To have a complete description of the protocol, one should also consider the case where A is willing to talk to B, B is willing to talk to I and symmetrically, all cases where A plays the role of B and B plays the role of A.

The ability of the intruder to analyze and forge new messages can be represented by the following set of clauses $\mathcal{C}_I$. It is the simple translation of the deduction system of Figure 1.

$$
\begin{array}{rr}
I(x), I(y) \Rightarrow I(\langle x, y\rangle) & I(x), I(y) \Rightarrow I(\mathsf{sign}(x, y)) \\
I(x), I(y) \Rightarrow I(\mathsf{enc}(x, y)) & I(x), I(y) \Rightarrow I(\mathsf{enca}(x, y)) \\
I(\langle x, y\rangle) \Rightarrow I(x) & I(\langle x, y\rangle) \Rightarrow I(y) \\
I(\mathsf{enc}(x, y)), I(y) \Rightarrow I(x) & I(\mathsf{enc}(x, \mathsf{pub}(y))), I(\mathsf{priv}(y)) \Rightarrow I(x) \\
I(\mathsf{enc}(x, \mathsf{priv}(y))) \Rightarrow I(x) &
\end{array}
$$

Then security of a protocol is reduced to checking satisfiability of a set of clauses. For example, the confidentiality of the nonce N_b can be expressed by the satisfiability of the set of clauses $\mathcal{C}_{\mathsf{NS}} \cup \mathcal{C}_I \cup \{\neg I(n_b)\}$.

This modeling of protocols is the approach used for by the ProVerif tool [8, 10], which has been successfully used for analyzing many security protocols (see e.g. [1, 11]). Some decidable fragments of Horn clauses, well suited for protocols have been proposed in [13, 25].

3 Computational Approach

The abstraction of messages by terms and the limited adversary raise some questions regarding the security guarantees offered by such proofs, especially from the perspective of the computational model.

3.1 Brief Presentation of Computational Models

In computational models, messages that are exchanged are bit-strings and depend on a security parameter η which is used, for example to determine the length of random nonces. In contrast to symbolic models, the attacker does not perform predetermined actions for analyzing messages, but is modeled by any probabilistic Turing machine running in polynomial-time w.r.t. the security parameter.

Security properties are also stated in a stronger way than in symbolic models. For example, the confidentiality of a nonce does not only say that an attacker should not be able to output the nonce but also require that the attacker should not be able to get any partial information about the nonce. Formally, confidentiality is expressed through a game. The game is parametrized by a bit b and involves an adversary $\mathcal{A}$. The input to the game is the security parameter η. It starts by generating two random nonces n_0 and n_1. Then the adversary $\mathcal{A}$ starts interacting with the protocol Π. It generates new sessions, sends messages and receives messages to and from these sessions (as prescribed by the protocol). At some point in the execution the adversary initiates a session and declares this session under attack. Then, in this session, the confidential nonce is instantiated with n_b (*i.e.* one of the two nonces chosen in the beginning of the experiment, the selection being made according to the bit b) and the adversary continues its interaction with the protocol. In the end, the adversary is given n_0 and n_1 and outputs a guess d. The nonce is computationally secret in Π if the probability that $d = b$ is the same than the probability that $d \neq b$ up to some negligible function[1] in the security parameter.

Under the computational approach, the security of protocols is based on the security of the underlying primitives, which in turn is proved assuming the hardness of solving various computational tasks such as factoring or taking discrete logarithms. The main tools used for proofs are *reductions*: to prove a protocol secure one shows that a successful adversary against the protocol can be efficiently

[1] A function f is said to be negligible if it grows slower than the inverse of any polynomial, that is, for any polynomial P, there exists n_0 such that for any $n \geq n_0$, $|f(n)| \leq \frac{1}{P(n)}$.

transformed into an adversary against some primitive used in its construction. Here, quantification is universal over *all* possible probabilistic polynomial-time (probabilistic polynomial time) adversaries and the execution model that is analyzed is specified down to the bit-string level. Two important implications stem from these features: proofs in the computational model imply strong guarantees (security holds in the presence of an *arbitrary* probabilistic polynomial-time adversary). At the same time however, security reductions for even moderately-sized protocols become extremely long, difficult, and tedious.

3.2 Bridging the Gap Between Symbolic and Computational Models

Recently, a significant research effort aims at linking the two approaches via *computational soundness* theorems for symbolic analysis [2, 6, 5, 4, 17]. Justifying symbolic proofs with respect to standard computational models has significant benefits: protocols can be analyzed and proved secure using the simpler, automated methods specific to the symbolic approach, yet the security guarantees are with respect to the more comprehensive computational model.

For example, it has been shown in [15] that for protocols with asymmetric encryption and signatures, any trace execution obtained by the interaction of a concrete (computational) adversary is (with overwhelming probability) the image of a symbolic execution trace obtained by executing a symbolic adversary. It is also proved that symbolic secrecy implies the computational (indistinguishability based) secrecy. This statement holds under standard assumptions on the security of the cryptographic primitives used in the concrete implementation, namely provided that encryption is IND-CCA2 secure and that signature is existentially unforgeable. Intuitively, IND-CCA2 security is defined through a game where the adversary should not be able to link cypher-texts to corresponding plain-texts even if he is given access to encryption and decryption oracles. A signature is existentially unforgeable whenever an adversary cannot produce valid signatures even being given access to a signature oracle.

One consequence of this result is that in the concrete model, the actions of an adversary are limited to the actions of the symbolic adversary. This allows to transfer trace-based security properties such as authentication and secrecy properties from symbolic to computational models. In other words, as soon as a protocol is proved secure for an authentication and secrecy property in symbolic models (using e.g. an automatic tool) then it is deemed secure in the less abstract computational model. This kind of results have then been extended e.g. to hash functions (in the random oracle model) [14], non-malleable commitment [19] and zero-knowledge proofs [7].

4 Conclusion

Symbolic approaches have proved their usefulness for analyzing security protocols. Automatic tools have been often used for discovering previously unknown

flaws. Abstracting messages by terms seems to be a good level of abstraction since it is possible to show that security proof in symbolic models actually implies stronger guarantees in computational models under classical assumptions under the implementation of the primitives.

There are still several open directions of research. Symbolic approaches currently allow to check classical security properties such as confidentiality and authentication. For more recent protocols such as e-voting protocols and contract-signing protocols, the properties that should be achieved are more involved and cannot be encoded in existing tools. In addition, these recent protocols make use of less classical primitives such as re-randomizable encryption scheme or blind signatures. New decision techniques have to be developed for these particular primitives and security properties.

Bridging the gap between symbolic and computation models is a promising line a research since it enables to prove strong security guarantees, benefiting from the simplicity of symbolic models. However, current results require strong assumptions on the security of the cryptographic primitives (e.g. IND-CCA2 encryption schemes). Weaker security assumptions like IND-CPA secure encryption schemes may not suffice to ensure security of protocols [26]. Using weaker encryption schemes may thus require to adapt both symbolic models and protocols accordingly.

References

1. M. Abadi, B. Blanchet, and C. Fournet. Just fast keying in the pi calculus. *ACM Transactions on Information and System Security (TISSEC)*, 10(3):1–59, July 2007.
2. M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 2:103–127, 2002.
3. R. Amadio and W. Charatonik. On name generation and set-based analysis in the dolev-yao model. In *Proc. of the 13th International Conference on Concurrency Theory (CONCUR'02)*, LNCS, pages 499–514. Springer Verlag, 2002.
4. M. Backes and B. Pfitzmann. Limits of the cryptographic realization of dolev-yao-style xor. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS'05)*, Lecture Notes in Computer Science, pages 336–354, 2005.
5. M. Backes and B. Pfitzmann. Relating cryptographic und symbolic key secrecy. In *26th IEEE Symposium on Security and Privacy*, pages 171–182, Oakland, CA, 2005.
6. M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations (extended abstract). In *Proc. of 10th ACM Conference on Computer and Communications Security (CCS'05)*, pages 220 – 230, 2003.
7. M. Backes and D. Unruh. Computational soundness of symbolic zero-knowledge proofs against active attackers. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 255–269, Pittsburgh, PA, USA, June 2008. IEEE Computer Society Press.
8. B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press, June 2001.

9. B. Blanchet. From secrecy to authenticity in security protocols. In *9th International Static Analysis Symposium (SAS'02)*, volume 2477 of *LNCS*, pages 242–259. Springer-Verlag, September 2002.
10. B. Blanchet. An automatic security protocol verifier based on resolution theorem proving (invited tutorial). In *20th International Conference on Automated Deduction (CADE-20)*, Tallinn, Estonia, July 2005.
11. B. Blanchet and A. Chaudhuri. Automated formal analysis of a protocol for secure file sharing on untrusted storage. In *IEEE Symposium on Security and Privacy*, pages 417–431, Oakland, CA, May 2008. IEEE.
12. B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In A. Gordon, editor, *Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *LNCS*, April 2003.
13. H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proc. of the 14th Int. Conf. on Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *LNCS*, pages 148–164. Springer-Verlag, 2003.
14. V. Cortier, S. Kremer, R. Küsters, and B. Warinschi. Computationally sound symbolic secrecy in the presence of hash functions. In N. Garg and S. Arun-Kumar, editors, *Proceedings of the 26th Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume 4337 of *Lecture Notes in Computer Science*, pages 176–187, Kolkata, India, December 2006. Springer.
15. V. Cortier and B. Warinschi. Computationally Sound, Automated Proofs for Security Protocols. In *Proc. 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171, Edinburgh, U.K, April 2005. Springer.
16. V. Cortier and E. Zalinescu. Deciding key cycles for security protocols. In *Proc. of the 13th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06)*, volume 4246 of *Lecture Notes in Artificial Intelligence*, pages 317–331, Phnom Penh, Cambodia, November 2006. Springer.
17. A. Datta, A. Derek, J. C. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic Polynomial-time Semantics for a Protocol Security Logic. In *Proc. of 32nd International Colloquium on Automata, Languages and Programming, ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2005. Lisboa, Portugal.
18. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. of the Workshop on Formal Methods and Security Protocols*, 1999.
19. D. Galindo, F. D. Garcia, and P. van Rossum. Computational soundness of non-malleable commitments. In *Proc. 4th Information Security Practice and Experience Conference (ISPEC'08)*, 2008. To appear.
20. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, march 1996.
21. G. Lowe. Casper: A compiler for the analysis of security protocols. In *Proc. of 10th Computer Security Foundations Workshop (CSFW'97)*, Rockport, Massachusetts, USA, 1997. IEEE Computer Society Press. Also in Journal of Computer Security, Volume 6, pages 53-84, 1998.
22. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. of the 8th ACM Conference on Computer and Communications Security (CCS'01)*, 2001.

23. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communication of the ACM*, 21(12):993–999, 1978.
24. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299:451–475, 2003.
25. H. Seidl and K. N. Verma. Flat and one-variable clauses: Complexity of verifying cryptographic protocols with single blind copying. In *Logic for Programming and Automated Reasoning (LPAR'04)*, LNCS, pages 79–94. Springer-Verlag, 2005.
26. B. Warinschi. A computational analysis of the Needham-Schroeder protocol. In *Proc. 16th IEEE Computer Science Foundations Workshop (CSFW'03)*, pages 248–262, 2003.