Challenges in the Automated Verification of Security Protocols

Hubert Comon-Lundh[1]

Ecole Normale Supérieure de Cachan and
Research Center for Information Security,
National Institute of Advanced
Industrial Science and Technology (AIST)
Tokyo, Japan
h.comon-lundh@aist.go.jp

Abstract. The application area of security protocols raises several problems that are relevant to automated deduction. We describe in this note some of these challenges.

1 Introduction

Security protocols are small distributed programs, aiming at securely achieving some transaction, while relying on a public or a non-reliable network. Getting formal guarantees that such protocols satisfy their specifications is an important issue, because of the numerous applications and the economical and societal impact of potential failures.

Program testing is irrelevant in this context, because of the malicious nature of attacks. On the opposite, security protocols constitute an ideal field of applications for research in automated deduction: the protocols are small, as well as their specification, yet their verification is non-trivial. The purpose of this note is to show several interesting issues in automated deduction, which are raised by this verification problem.

We will focus on problems that can (or could) be formulated in first-order logic or equational logic. Already in early works, first-order clausal formalisms have been used for security protocols [45] and intruder capabilities formalized using term rewriting systems [47]. Since then, there has been a large amount of work in the area, which *we do not survey here* (in particular many relevant references are certainly missing). Despite this research, there are still open questions, some of which are described in this note.

We will organize the note stepwise, starting with the most elementary notions (intruder deduction) in section 2 and enrich the model in further sections, each time reporting some unsolved problem. In section 3, we consider the security problem for a fixed number of protocol sessions. In section 4, we consider resolution-based theorem proving applied to security protocols. In section 5, we consider security properties that are usually defined as indistinguishability between two processes. Finally, we consider modularity issues in section 6.

2 Intruder deductions

The security of a protocol depends on several inputs: the protocol, the security property, but also the intruder capabilities and the (assumed) algebraic properties of the security primitives. We assume that $\mathcal{F}$ is a set of function symbols representing all security primitives and data. $\mathcal{A}$ is a set of equations, typically associativity and commutativity of some symbols in $\mathcal{F}$ and $\mathcal{R}$ is a rewrite system, convergent modulo $\mathcal{A}$ capturing the expected computations on messages. A typical rule in $\mathcal{R}$ would be $\mathsf{dec}(\mathsf{enc}(x, y), y^{-1}) \to x$.

In the simplest setting, we only consider offline attacks: we forget how an attacker got messages and simply assume (s)he holds a set of messages. In addition, we only consider security properties that can be modeled as the confidentiality of some piece of a message.

We also assume here that the security primitives, such as encryption algorithms, do not leak any information besides what is modeled through the algebraic properties: we consider a formal model, that is sound with respect to the computational one.

In such a (restricted) case, security is an *Entscheidungsproblem* for a particular deduction system, representing the attacker computing capabilities. We let $\mathcal{F}_{\text{pub}}$ be the subset of $\mathcal{F}$ consisting of *public symbols*. Messages are represented as terms in the algebra $T(\mathcal{F})$ (or possibly a quotient $T(\mathcal{F})/_{=_{\mathcal{A}}}$) and the inference rules are of the form

$$\frac{s_1 \ \ldots \ s_n}{f(s_1, \ldots, s_n)\downarrow} R_f$$

The attacker may apply the function $f \in \mathcal{F}_{\text{pub}}$ to messages $s_1, \ldots, s_n$, getting $f(s_1, \ldots, s_n)\downarrow$, the normal form of $f(s_1, \ldots, s_n)$ with respect to $\mathcal{R}$.

Example 1. The most typical example is the *perfect cryptography*, for symmetric encryption. $\mathcal{F}_{\text{pub}}$ contains pairing: $< s_1, s_2 >$ is the pair of the two terms s_1, s_2, encryption: $\{s\}_k$ is the encryption of s with k, decryption dec, projections π_1, π_2 and some public constants. $\mathcal{F} = \mathcal{F}_{\text{pub}} \cup \mathcal{S}$ where $\mathcal{S}$ is a set of (secret) constants, disjoint from $\mathcal{F}_{\text{pub}}$. $\mathcal{A}$ is empty and the rewrite system consists of three rules:

$$\mathsf{dec}(\{s\}_k, k) \to s \qquad \pi_1(< x, y >) \to x \qquad \pi_2(< x, y >) \to y$$

Many examples of (other) relevant equational theories and rewrite systems are given in [33].

The set of inference rules needs not be finite, as, for variadic symbols (or associative-commutative symbols), we may want one version of the rule for each value of n. It is also convenient sometimes to consider combinations of function symbols instead of a single function application.

Also note the resulting term $f(s_1, \ldots, s_n)\downarrow$ is assumed to be a message: the set of messages M is a subset of $T(\mathcal{F})$ (or $T(\mathcal{F})/_{=_{\mathcal{A}}}$) and a valid instance of the rule assumes that both the premisses and the conclusion are in M. For instance, in example 1, terms in M do not contain the symbols $\mathsf{dec}, \pi_1, \pi_2$ and, if the decryption does not succeed, the attacker does not learn anything about the plaintext.

The set of inference rules yields a deducibility relation $\vdash \subseteq 2^M \times M$, using repeatedly the inference rules from an initial knowledge $H \subseteq 2^M$.

In most relevant cases, as shown in [29], an equivalent inference system can be obtained by "inlining" the rewriting rules applications, getting rid of the normalization symbol; the rewriting steps in the conclusion are replaced with finitely many patterns in the premisse. In example 1, we would get the following inference rules:

$$\frac{x_1 \quad x_2}{\{x_1\}_{x_2}} \qquad \frac{x_1 \quad x_2}{< x_1, x_2 >} \qquad \frac{\{x_1\}_{x_2} \quad x_2}{x_1} \qquad \frac{< x_1, x_2 >}{x_1} \qquad \frac{< x_1, x_2 >}{x_2}$$

Definition 1. *Given an intruder deduction system, the* deducibility problem *is, given a finite subset H of M and a term $s \in M$, to decide whether $H \vdash s$.*

This may also be formulated using rewriting systems: if we let $H = \{t_1, \ldots, t_n\}$ and σ be the substitution $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, the problem is equivalent to:

Given σ, is there a term $\zeta \in T(\mathcal{F}_{\text{pub}}, \{x_1, \ldots, x_n\})$ such that $\zeta\sigma \downarrow =_{\mathcal{A}} s$.

Now, what is a bit surprising at first sight is that many of such intruder deduction systems have very nice properties: the deducibility problem is in PTIME. This relies on a subformula property of the deduction system: if the function symbols applications are gathered together in an appropriate way, then for any deducible s, there is a proof in which intermediate steps belong to $\mathsf{St}(s, H)$, the subterms of the hypotheses and the conclusion. In such a case, we say that the deduction system is *local*, following [44]. Then, if one step deduction is itself decidable in PTIME, the deducibility problem can be reduced to a simple fixed point computation of the deducible subterms.

Such a locality theorem has been established for many examples of intruder deduction systems, for instance perfect cryptography, for symmetric and asymmetric encryption and signatures, commutative encryption, ECB and CBC encryption modes, blind signatures, exclusive or [22], Abelian groups [21], modular exponentiation, and others [35, 19]. Actually, each time a new protocol is designed, which relies on a new equational theory, a new locality result is established.

Instead of repeating the work for each new theory, we wish to get a more general result. This raises the following challenge:

Challenge 1 *For which classes of (AC)-rewrite systems, do we get a locality property? A decidable deducibility problem?*

Related works

There are actually some works [30, 5], answering, at least partly, this question, *when $\mathcal{A}$ is empty*. Many examples however involve associative and commutative symbols. So, the above challenge can be seen as extending these results to the AC case.

[8] also provides with a general criterion, amenable to automated locality proofs. First, locality is generalized, replacing the subterm relation with any well-founded ordering: $\mathsf{St}(s, H)$ above is replaced with the set of terms that are smaller than s, H. Of course, if we want to derive a decision procedure, this set must be finite. Next, they show that non-local inference systems can be completed into local ones, using ordered resolution (*saturated sets* are local). However, this process may not terminate. That is what happens when, in our applications, $\mathcal{A}$ is not empty. Typically, we need to deal with infinitely many clauses of the form $I(x_1), \ldots, I(x_n) \rightarrow I(x_1 + \ldots + x_n)$ when $+$ is associative and commutative. Answering the challenge along this line of research would require to extend the results of [8] to some class of infinite sets of clauses.

Other interesting issues

Another related problem is the *combination* of intruder theories: If the deducibility problem is decidable for two intruder theories, under which condition is it decidable in the union of the two theories ? Similar questions may be asked for local theories.

This is an important issue when the deduction system is large. For instance in [19], the authors consider an intruder deduction system, which includes several associative-commutative symbols and a lot of rules. This system formalizes some arithmetic properties, which must be considered in the model, since they are used by the participants of an actual electronic purse protocol. The locality proof of [19] is a direct (tedious) proof and would benefit from combination results.

There is a combination result in the disjoint case [7], however not applicable to the above example.

3 Bounded number of sessions

Now, we enrich the model, considering some active attacks: the attacker may not only perform offline deductions, but also send fake messages and take advantage of the reply. We assume however that he can send only a fixed number of such messages (this number is given as a parameter).

One of the main problems, which yields "man-in-the-middle attacks" such as the famous attack of [42], is that pieces of messages cannot be analysed by some honest parties, hence could be replaced by arbitrary messages.

Example 2. An honest agent a generates a random number n and sends it, together with his identity, encrypted with the public key of another honest agent b. Such a message can be represented as $\{< a, n >\}_{\mathsf{pub}(b)}$, a (ground) term in our algebra $T(\mathcal{F})$. However, b's view is different. Upon receiving this message, (s)he decrypts and finds a pair, whose second component is an arbitrary message. Suppose that b has to send back $\{< b, n >\}_{\mathsf{pub}(a)}$, the corresponding rule for b will be:

Upon receiving any $\{< x, y >\}_{\mathsf{pub}(b)}$ reply by sending $\{< b, y >\}_{\mathsf{pub}(x)}$,

maybe after checking that x is the identity of somebody with whom b is willing to communicate.

The important issue here is that y can be replaced by any message and the attacker may take advantage of this.

Hence, a simple class of protocols can be defined by finitely many *roles*, which are again finite sequences of rules $s \Rightarrow t$, meaning that, upon reception of a message $s\sigma$, the agent replies $t\sigma$. In general, each role also generates new names (keys or nonces) and may have a more complex control structure, for instance specifying what to do when the message does not match the expected s. In this section, we will not consider replication of the roles, hence new names can simply be distinguished using distinct constants and we only consider a simple linear structure of the control. Even more, by guessing an ordering between the different actions of each role, we get a sequence $s_i \Rightarrow t_i$ of query $\Rightarrow$ response (such as $\{< x, y >\}_{\mathsf{pub}(b)} \Rightarrow \{< b, y >\}_{\mathsf{pub}(x)}$ in example 2),

In this setting, there is an attack on the secrecy of s, if there is a substitution σ such that

$$\begin{aligned} S_0 &\vdash s_1\sigma \\ S_0, t_1\sigma &\vdash s_2\sigma \\ &\vdots \\ S_0, t_1\sigma, \ldots, t_{n-1}\sigma &\vdash s_n\sigma \\ S_0, t_1\sigma, \ldots, t_n\sigma &\vdash s \end{aligned}$$

where S_0 is the set of initial intruder knowledge and s is the data supposed to remain secret. $s_1\sigma, \ldots, s_n\sigma$ are the successive (fake) messages constructed by the attacker and $t_1\sigma, \ldots, t_n\sigma$ are the successive messages sent by the agents. The left members of the deducibility constraints increase as the attacker's gets new messages from the network. Without the last constraint, there is a trivial substitution σ_0, corresponding to the normal execution of the protocol, in which the attacker simply forwards the messages without modifying them : $s_{i+1}\sigma_0 = t_i\sigma_0$ for every $i < n$.

Equivalently, we can define an attack using term rewriting systems: there is an attack if there are terms $\zeta_0, \ldots, \zeta_n \in T(\mathcal{F}_{\text{pub}} \cup S_0, \{x_1, \ldots, x_n\})$ and a substitution σ such that

- For every $0 \leq i \leq n-1$, $(\zeta_i\{x_1 \mapsto t_1, \ldots, x_i \mapsto t_i\})\sigma\downarrow =_{\mathcal{A}} s_{i+1}\sigma\downarrow$
- $(\zeta_n\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\})\sigma\downarrow =_{\mathcal{A}} s$

$\zeta_0, \ldots, \zeta_n$ are the *recipes*, which constitute the attacker's strategy: they specify how to combine the known messages to produce the next fake message.

Deciding if there is an attack consists then in deciding whether such a substitution σ exists. This problem has been shown to be in NP for a number of intruder theories [49, 23, 22, 21, 46].

There are also a number of results for classes of theories, for instance: subterm theories [11] and monoidal theories [37]. This does not cover however some relevant examples such as the one described in [19].

In most (if not all) these results, the key is a *small attack property* (also called *conservativity* in e.g. [37]): if there is an attack then there is an attack,

which is built, stacking pieces of protocol rules. Then, the search space for σ becomes finite, yielding decision algorithms.

This property is not well-understood; the existence of small attacks can be recasted as the existence of particular (small) proofs: there should be proof simplification rules, which justify such a property. This yields the following challenge:

Challenge 2 *Give a proof-theoretic explanation of the small attack property.*

Related work

A partial answer is given in [14]: the authors give proof simplification rules, which are strongly normalizing. The normal proofs only correspond to small attacks. However, this work only covers a restricted class of intruder deduction rules, assuming $\mathcal{A}$ is empty, as well as additional syntactic restrictions.

An approach along the lines of [8] does not work here: the clausal formalisation of the protocol rules introduces some over-approximations, as explained in the next section.

Other interesting issues

Again, combination of decision procedures for a bounded number of sessions is an important issue, as, in general, many protocols are running at the same time on the same device. This problem is addressed in [24, 25], but there are still examples of relevant complex equational theories that cannot (and should) be broken into pieces using these results (see e.g. [19]). Hence, there is still some work to be done in this area. (See also section 6 for the general modularity problem, when the number of sessions is unbounded.)

4 Clausal theorem proving and security protocols

In the automated deduction community, it has become a habit to recast all decision results as terminating resolution strategies [39]. This allows in principle to use a single clausal theorem prover for any of the particular decision problems. What about the decision results of the last section? This will be our next challenge, but let us first introduce and clarify the problems.

There are several formalizations of security protocols within first-order logic. After C. Meadows [45], C. Weidenbach in [50] was one of the first who observed that resolution methods are quite successful in this area. Then the security analyzer ProVerif [15, 16] is also a resolution-based prover, and one of the most successful protocol verifiers.

Such formalizations rely on approximations. Otherwise, an accurate model such as [28] is hopeless, as far as termination is concerned. Among the approximations:

– Nonces, i.e., new names that are generated at each session of the protocol, are represented as terms, depending on the environment in which they are generated. This over-approximation is always correct as long as secrecy properties are considered. As already mentioned, for a bounded number of sessions, we can simply represent these nonces using different names.
– The ordering in which messages of a given role are played is lost in the clausal translation. This can be recovered in case of a bounded number of sessions, recording and verifying the ordering of events.
– Clauses are universally quantified formulas. This is fine for clauses representing the attacker's capabilities, such as

$$I(x), I(y) \rightarrow I(< x, y >)$$

as the attacker may re-use such a construction any number of times. This might be more problematic for protocol rules as, when we use variables in place of messages (for un-analysable parts), the attacker may choose any value, but has to commit to this value: the rule should not be replayed.

This last problem is well-known in automated deduction (typically in tableau methods): we need *rigid variables*. While rigid variables are still universally quantified, they can get only one instance: each rigid clause can be instanciated only once and then used as many times as we wish. Using a mixture of rigid clauses and flexible clauses is therefore more accurate for the security protocols verification. This has been investigated in [38].

We may however simply translate the rigid clauses into first-order clauses, at the price of adding new arguments, while preserving satisfiability [4]. We get in this case a faithful translation into first-order clauses of the security problem for a bounded number of sessions.

Challenge 3 *Design a resolution strategy, which yields a NP-decision method for the formulas corresponding to bounded sessions protocol security.*

Related work

Coming back to the over-approximated problem for an unbounded number of sessions, resolution methods are unusually successful: in many cases, the deduction process terminates. There are several works explaining this phenomenon. For instance, restrictions on the variables occurring in each protocol clause, the protocol model falls into a decidable fragment of first-order logic [34, 27, 26]. Also, assuming some tagging of (sub)messages yiedls termination [18, 48].

Other interesting issues

They include speeding-up the deduction, relying on constraints [9] or strategies [16] or including general (user-defined) equational theories [10, 29, 17].

5 Proofs of equivalences

Many security properties cannot be directly modeled in first-order logic, since they are not properties of a single execution trace. They are better stated as indistinguishability properties between two processes. Typical such examples are anonymity and (strong) secrecy; while secrecy (or confidentiality) requires a given message s to remain unknown to the attacker, strong secrecy requires that an attacker cannot learn anything about s. The latter is expressed by an indistinguishability property between the real protocol and the protocol in which s is replaced by a new name. Properties of electronic voting protocols are also typically expressed in this way [36].

In case of equivalence properties, the deduction problem (definition 1) is replaced by deciding *static equivalence*:

Definition 2. *Two sequences of ground terms $s_1, \ldots, s_n$ and $t_1, \ldots, t_n$ are* statically equivalent*, which we write $s_1, \ldots, s_n \sim t_1, \ldots, t_n$, if, for any terms $u, v \in T(\mathcal{F}, \{x_1, \ldots, x_n\})$,*

$$\begin{array}{c} u\{x_1 \mapsto s_1, \ldots, x_n \mapsto s_n\}\downarrow =_{\mathcal{A}} v\{x_1 \mapsto s_1, \ldots, x_n \mapsto s_n\}\downarrow \\ \Updownarrow \\ u\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}\downarrow =_{\mathcal{A}} v\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}\downarrow \end{array}$$

The decision of static equivalence has been shown for several term rewriting systems [1, 32] and has been implemented for a wide class of equational theories [13, 17].

Consider now protocols. Each role can be described by a simple sequential process, for instance in the applied π-calculus [2] and each protocol is expressed as a parallel composition of a fixed number of roles $P_1, \ldots, P_n$ (resp. $Q_1, \ldots, Q_n$). The security properties that we wish to consider are expressed as the observational equivalence

$$!(P_1 \| \cdots \| P_n) \approx !(Q_1 \| \cdots \| Q_n)$$

In words: an attacker cannot distinguish between the two versions of the protocol. The exclamation mark is a replication: we may consider any number of copies of each protocol roles.

In the bounded sessions version of the problem, we simply remove the exclamation marks, and we can replace all fresh names by distinct constants. We call *bounded protocols* the resulting processes. As before, any interleaving τ_P of actions of the process $P = P_1 \| \cdots \| P_n$ corresponds to a sequence of query $\Rightarrow$ response: $s_i \Rightarrow t_i$, for $i = 1, ..., m$. For any such sequence, a *valid attacker on τ_P* consists in a substitution σ and terms $\zeta_0, \ldots, \zeta_{m-1} \in T(\mathcal{F}_{\text{pub}} \cup S_0, \{x_1, \ldots, x_{m-1}\})$ such that, for every i, $\zeta_i\theta_i\sigma\downarrow =_{\mathcal{A}} s_{i+1}\sigma\downarrow$, where θ_i is the substitution $\{x_1 \mapsto t_1, \ldots, x_i \mapsto t_i\}$ (the empty substitution when $i = 0$). This matches our definitions in section 3: $\zeta_0, \ldots, \zeta_{m-1}$ are the recipes specifying the attacker's strategy.

Now, the bounded protocols P and Q are *equivalent* if, for any interleaving τ_P of actions of P, for any valid attacker on τ_P $(\zeta_0, \ldots, \zeta_{m-1}, \sigma)$, there is an

interleaving τ_Q of actions of Q and a substitution σ' such that $(\zeta_0, \ldots, \zeta_{m-1}, \sigma')$ is a valid attacker on τ_Q and $t_1\sigma, \ldots, t_m\sigma \sim t'_1\sigma', \ldots, t'_m\sigma'$ where $t_1, \ldots, t_m$ is the sequence of responses in τ_P and $t'_1, \ldots, t'_m$ is the sequence of responses in τ_Q (and conversely, exchanging P and Q). In words: P and Q are equivalent if, whatever the attacker does, the sequence of replies from P is statically equivalent to the corresponding sequence of replies of Q.

All decision results reported in section 3 only concern the secrecy property. We would like to extend them to more security properties:

Challenge 4 *In case of a bounded number of sessions, give decision and complexity results for the equivalence of protocols.*

The problem is more complex than in the secrecy case, as we cannot guess first a trace and then solve a finite deducibility constraint system.

Related work

As far as we know, the only related works are

- a decision result for the spi-calculus (and no equational theory) [40] and
- a decision result for an approximate equivalence (implying observational equivalence) and subterm-convergent theories [12].

Other interesting issues

If we consider an arbitrary number of sessions, in [17], the authors design a coding of pairs of processes into *bi-processes*. In ProVerif, this coding itself is translated into first order-clauses, yielding automatic proofs of equivalences, for an unbounded number of sessions. To the best of our knowledge, this is the only tool, which allows for automatic equivalence proofs for an unbounded number of sessions. The encoding into bi-processes requires however an approximation of bisimulation relations: two processes might be equivalent and yet there is no equivalence proof in ProVerif. An interesting open question is to find a class of protocols/properties for which the bi-process technique is complete.

6 Modularity

The problem of combining protocols is an important issue. Usually, several protocols may run on the same device and it is important that, once each single protocol is proved secure, they remain secure when they run concurrently.

There are more reasons for studying this problem of security proofs modularity, which are coming from another area of computer science. In the area of computational security, usually related to cryptology, a huge amount of work has been devoted to *universal composability* [20, 41]. The idea is to get modularity theorems (called *composition theorems* in this context), which show that the security for one session implies the security for an unbounded number of

sessions and that two secure protocols can run concurrently, without generating new attacks. However, both the composition theorems and the proofs that security primitives satisfy the hypotheses of these theorems are quite delicate in the computational world [41]. Even more, to the best of our knowledge, there is no composition result in the computational setting, which would allow some secrets (such as a pair <public key, private key>) to be shared by protocols. Using a single public key in several protocols is however a common practice (and actually desirable).

In order to get computational security guarantees, there is an alternative method: using *soundness theorems* in the style of [3], the computational models are faithfully abstracted into symbolic ones. Then, proving the security at the computational level is reduced to proving a corresponding appropriate property at the symbolic level, which might be automated. There is a large body of literature on soundness theorems, which we do not review here.

Assuming appropriate soundness results, the modularity of symbolic security proofs would have therefore an important impact in the computational security community.

To be more precise, a modularity result in the symbolic setting would state that for a family of shared names $\overline{n}$ (typically the public and private keys), a class of protocols, a class of intruder deduction systems and a class of security properties, the security of two individual protocols P_1 and P_2 implies the security of the combined protocol $P_1 \| P_2$.

Challenge 5 *For which intruder deduction systems and security properties can we get modularity theorems?*

Additionally, for applications to universal composability, but also in order to get decision results, we wish to get some kind of modularity of a protocol with itself: assuming that secrets such as pairs of public and private keys are shared over sessions, the security of P should imply the security of $!P$.

Challenge 6 *For which classes of protocols, intruder theories and security properties does the security for a single session imply the security for an arbitrary number of sessions?*

Related work

A recent result [31] proves a modularity theorem for a wide class of protocols. However, it only considers the secrecy property and the intruder deduction system is fixed to the simple rules of example 1 for symmetric and asymmetric encryption. Generalizing the result to equivalence properties and other intruder systems is an open question.

The proof of this result relies on several techniques from automated deduction:

- The small attack property (see section 3),
- Properties of unification and deducibility constraints,

– An analog of Craig's interpolation theorem for intruder deduction systems.

Concerning the challenge 6, a first result was proved by G. Lowe [43], however for a restricted class of protocols. For instance [43] assumes that any two distinct terms headed with encryption, that occur in the protocol, are not unifiable. A similar result is shown in [6] for another class of protocols. However, in both cases, only secrecy is considered and the simple intruder deduction system is the one of example 1 (for symmetric and asymmetric encryption).

Acknowledgments

Many thanks to Reynald Affeldt, David Basin, Bruno Blanchet, Véronique Cortier, Stéphanie Delaune, Michael Rusinowitch for their comments on an early version of this note.

References

1. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 367(1–2):2–32, 2006.
2. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, January 2001.
3. M. Abadi and P. Rogaway. Reconciling two views of cryptography: the computational soundness of formal encryption. In *Proc. 1rst IFIP International Conference on Theoretical Computer Science*, volume 1872 of *Lecture Notes in Computer Science*, Sendai, Japan, 2000.
4. R. Affeldt and H. Comon-Lundh. First-order logic and security protocols. unpublished manuscript, Apr. 2008.
5. S. Anantharaman, P. Narendran, and M. Rusinowitch. Intruders with caps. In *Proc. 18th Int. Conf. on Rewriting Techniques and Applications (RTA)*, volume 4533 of *Lecture Notes in Computer Science*, 2007.
6. M. Arapinis, S. Delaune, and S. Kremer. From one session to many: Dynamic tags for security protocols. Research Report LSV-08-16, Laboratoire Spécification et Vérification, ENS Cachan, France, May 2008.
7. M. Arnaud, V. Cortier, and S. Delaune. Combining algorithms for deciding knowledge in security protocols. In *Proc. Int. Symp. on Frontiers of Combining Systems (FroCoS)*, volume 4720 of *Lecture Notes in Artificial Intelligence*. Springer, 2007.
8. D. Basin and H. Ganzinger. Automated complexity analysis based on ordered resolution. *Journal of the Association of Computing Machinery*, 48(1):70–109, January 2001.
9. D. Basin, S. Mödersheim, and L. Viganò. Constraint Differentiation: A New Reduction Technique for Constraint-Based Analysis of Security Protocols. In *Proceedings of CCS'03*, pages 335–344. ACM Press, New York, 2003.
10. D. Basin, S. Mödersheim, and L. Viganò. Algebraic intruder deductions. In *Proc. Int. Conf. on Logic for Programming, and Automated Reasoning (LPAR)*, volume 3835 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2005.

11. M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 16–25, Alexandria, Virginia, USA, Nov. 2005. ACM Press.
12. M. Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, Jan. 2007.
13. M. Baudet. Yapa: Yet another protocol analyser, 2008. available at `http://www.lsv.ens-cachan.fr/~baudet/yapa/index.html`.
14. V. Bernat and H. Comon-Lundh. Normal proofs in intruder theories. In *Revised Selected Papers of the 11th Asian Computing Science Conference (ASIAN'06)*, volume 4435 of *Lecture Notes in Computer Science*. Springer, 2008.
15. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96, Cape Breton, Nova Scotia, Canada, June 2001. IEEE Computer Society.
16. 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.
17. B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.
18. B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. *Theoretical Computer Science*, 333(1–2):67–90, Mar. 2005.
19. S. Bursuc, H. Comon-Lundh, and S. Delaune. Deducibility constraints, equational theory and electronic money. In *Rewriting, Computation and Proof — Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of his 60th Birthday*, volume 4600 of *Lecture Notes in Computer Science*. Springer, 2007.
20. R. Canetti and T. Rabin. Universal composition with joint state. Cryptology ePrint Archive, report 2002/47, Nov. 2003.
21. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents. In *Proc. FST/TCS, Mumbai*, volume 2914 of *Lecture Notes in Computer Science*, 2003.
22. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS)*, 2003.
23. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with commuting public key encryption. In *Proc. Workshop on Automated Reasoning for Security Protocol Analysis (ARSPA), Electronic Notes in Theoretical Computer Science*, volume 125, 2004.
24. Y. Chevalier and M. Rusinowitch. Combining Intruder Theories. In *Proc. 32nd Int. Conference on Automata, Languages and Programming (ICALP)*, volume 3580 of *Lecture Notes in Computer Science*. Springer, 2005.
25. Y. Chevalier and M. Rusinowitch. Hierarchical combination of intruder theories. In *Proc. Rewriting Techniques and Application*, volume 4098 of *Lecture Notes in Computer Science*, pages 108–122, 2006.
26. H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. *Theoretical Computer Science*, 331(1):143–214, Feb. 2005.
27. H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *14th Int. Conf. Rewriting Techniques and Applications (RTA)*, volume 2706 of *Lecture Notes in Computer Science*. Springer, 2003.

28. H. Comon-Lundh and V. Cortier. Security properties: Two agents are sufficient. *Science of Computer Programming*, 50(1-3):51–71, Mar. 2004.
29. H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In *Proc. 16th Int. Conf. on Rewriting Techniques and Applications (RTA)*, volume 3467 of *Lecture Notes in Computer Science*, 2005.
30. 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*. Springer, 2003.
31. V. Cortier, J. Delaitre, and S. Delaune. Safely composing security protocols. In *Proc. 27th Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 4855 of *Lecture Notes in Computer Science*. Springer, 2007.
32. V. Cortier and S. Delaune. Deciding knowledge in security protocols for monoidal equational theories. In *Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 4790 of *Lecture Notes in Artificial Intelligence*. Springer, 2007.
33. V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.
34. V. Cortier, M. Rusinowitch, and E. Zalinescu. A resolution strategy for verifying cryptographic protocols with CBC encryption and blind signatures. In *7th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP)*, 2005.
35. S. Delaune. Easy intruder deduction problems with homomorphisms. *Information Processing Letters*, 97(6):213–218, Mar. 2006.
36. S. Delaune, S. Kremer, and M. D. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Proceedings of the 19th Computer Security Foundations Workshop (CSFW)*. IEEE Comp. Society Press, 2006.
37. S. Delaune, P. Lafourcade, D. Lugiez, and R. Treinen. Symbolic protocol analysis for monoidal equational theories. *Information and Computation*, 206:312–351, Feb.-Apr. 2008.
38. S. Delaune, H. Lin, and Ch. Lynch. Protocol verification via rigid/flexible resolution. In *Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 4790 of *Lecture Notes in Artificial Intelligence*. Springer, 2007.
39. C. Fermüller, A. Leitsch, U. Hustadt, and T. Tamet. Resolution decision procedure. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 2, chapter 25, pages 1793–1849. North Holland, 2001.
40. H. Huttel. Deciding framed bisimulation. In *4th International Workshop on Verification of Infinite State Systems INFINITY'02*, pages 1–20, 2002.
41. R. Küsters and M. Tuengerthal. Joint state theorems for public-key encryption and digital signature functionalities with local computations. In *Computer Security Foundations (CSF'08)*, 2008.
42. G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1996.
43. G. Lowe. Towards a completeness result for model checking of security protocols. *Journal of Computer Security*, 7(1), 1999.
44. D. McAllester. Automatic recognition of tractability in inference relations. *J. ACM*, 40(2), 1993.
45. C. Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.
46. J. Millen and V. Shmatikov. Symbolic protocol analysis with an Abelian group operator or Diffie-Hellman exponentiation. *J. Computer Security*, 2005.

47. J. K. Millen and H.-P. Ko. Narrowing terminates for encryption. In *Proc. Ninth IEEE Computer Security Foundations Workshop (CSFW)*, 1996.
48. R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable with unbounded nonces as well. In *Proc. Foundations of Software Technology and Theoretical Computer Science (FST-TCS)*, volume 2914 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2003.
49. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is np-complete. In *Proc.14th IEEE Computer Security Foundations Workshop*, Cape Breton, Nova Scotia, June 2001.
50. C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *Proc. 16th Conference on Automated Deduction*, volume 1632 of *Lecture Notes in Computer Science*, pages 314–328, 1999.