# Introduction

Véronique CORTIER [a] and Steve KREMER [b]
[a] *LORIA, CNRS*
[b] *LSV, ENS Cachan & CNRS & INRIA*

Formal methods have shown their interest when developing critical systems, where safety or security is important. This is particularly true in the field of security protocols. Such protocols aim at securing communications over a public network. Small flaws in the development of such systems may cause important economical damages. Examples of security protocols include the Transport Layer Security (TLS) protocol and its predecessor, the Secure Sockets Layer (SSL). These protocols are typically used for guaranteeing a secure connection to a web site in particular for secure payment over the Internet. Most web browsers display a small lock to indicate that you are executing a secure session using one of these protocols. Another emergent application of security protocol is electronic voting. For instance, in the 2007 national elections in Estonia the government offered the possibility to vote via the Internet. The development of such protocols is error-prone and flaws are regularly discovered. For example, the SAML 2.0 Web Browser Single Sign-On authentication system developed by Google has recently been attacked. The Single Sign-On protocol allows a user to identify himself only once and then access to various applications (such as Gmail or Google calendar). While designing a formal model of this protocol, Armando *et al* [ACC+08] discovered that a dishonest service provider could actually impersonate any of its users at another service provider. This flaw has been corrected since. Those examples show the need of precise security guarantees when designing protocols. Moreover, the relatively small size of security protocols makes the use of formal verification reasonable.

The use of symbolic methods for formally analyzing security protocols goes back to the seminal paper of Dolev and Yao [DY81]. While there is not a unique symbolic model, the so-called *Dolev-Yao models* generally share the following ingredients: the adversary is computationally unbounded and has complete control of the network while cryptography is assumed to be perfect. For example, the adversary is not allowed to perform cryptanalysis or to decrypt a ciphertext without knowing the decryption key. Finding collisions or guessing fresh nonces is also supposed to be impossible, *etc*. Most early tools [Mil84,Low96b] and techniques [BAN89] were aiming above all at finding bugs in protocols. Many errors have indeed been identified using formal methods, demonstrating their usefulness. At the end of the '90s more foundational questions were investigated: the general undecidability results for automated verification of security protocols have been refined and decidable classes of protocols and restrictions yielding decidability were identified together with their complexity [DLM04,CC01,RT01]. At about the same time, models [THG99,AF01] and tool support [Pau98,Bla01] were also developed for proving protocols correct rather than only finding flaws. When the focus shifted from finding flaws to proving security protocols correct, a natural question was raised about the guarantees provided in these models relying on the so-called perfect cryptography

assumption. A lot of efforts were performed to relax this assumption by introducing algebraic properties of cryptographic primitives (see [CDL06] for a survey) or proving that symbolic proofs can be transferred to more realistic, computational models starting with [AR00]. Investigating these foundational questions not only sharpened our understanding of the underlying difficulties of security protocol verification but also enabled the development of efficient tools such as among others the AVISPA platform [ABB+05], the ProVerif [Bla01] and the Scyther [Cre08] tools. In recent years there have also been works on widening the scope of the class of security protocols and properties that can be analyzed, going beyond the classical protocols for guaranteeing authentication and confidentiality. For instance the ProVerif tool allows to check the resistance against dictionary attacks [Bla04], as well as indistinguishability properties [BAF05]. Moreover, complex properties of contract signing [KR02,CKS01,KK05] and electronic voting protocols [DKR09,KT09,BHM08a] have been formalized.

The theory as well as the tools for formal analysis have now reached a state of maturity where they can be used on practical protocols. However, there is not one technique or tool which combines all benefits. There exist today many formalisms and approaches which have each their own benefits. The aim of this book is to give an overview of the state of the art of the field by showing some of the most influential developments in this field.

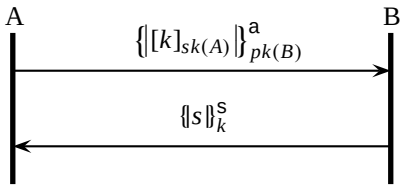
## 1. Some examples of security protocols

Security protocols aim at different goals such as key distribution, message integrity, authentication, non repudiation or voting. They all make use of cryptographic primitives as a key ingredient to achieve security. Popular cryptographic primitives are symmetric and asymmetric encryption, signatures, hash function and MACs for example. For the most common primitives, we introduce hereafter notations that will be used throughout all the chapters of this book. In particular, the symmetric encryption of a message $m$ with key $k$ is denoted by $\{\!|m|\!\}^{\mathsf{s}}_k$. Similarly, we write $\{\!|m|\!\}^{\mathsf{a}}_{pk}$ for the asymmetric encryption of $m$ with the public key $pk$ and $[m]_{sk}$ for the digital signature of $m$ with the secret key $sk$. $\langle m_1, m_2\rangle$ denotes the pairing of the two messages $m_1$ and $m_2$.

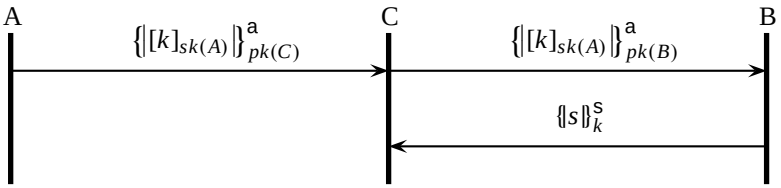
To illustrate the different results described in this book, we will use similar examples of protocols. This should allow the reader to compare the differences between each approach.

### *1.1. Handshake protocol*

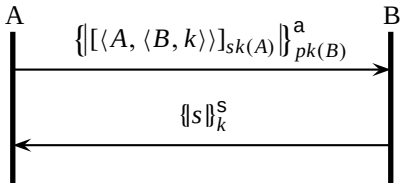
A first example of protocol is a naive handshake protocol between $A$ and $B$, illustrated in Figure 1(a). This protocol has been proposed for illustration purposes by Blanchet [Bla08]. The aim of the protocol is that $A$ and $B$ share a secret $s$ at the end. Participant $A$ generates a fresh session key $k$, signs it with his secret key $sk(A)$ and encrypts it using $B$'s public key $pk(B)$. When $B$ receives this message he decrypts it using his secret key, verifies the digital signature and extracts the session key $k$. $B$ uses this key to symmetrically encrypt the secret $s$. The rationale is that when $B$ receives this message he should be the only one able to know its content (because of the asymmetric encryption). Moreover, the digital signature should ensure that $A$ is the originator of the message.




(a) (Flawed) handshake protocol




(b) Man in the middle attack




(c) Corrected handshake protocol

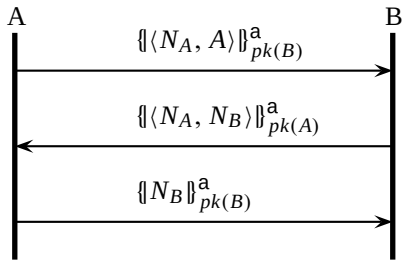
**Figure 1.** Handshake protocol

However, this protocol is vulnerable to a *man in the middle* attack described in Figure 1(b). If $A$ starts a session with a dishonest participant $C$ then $C$ is able to impersonate $A$ in a session he starts with $B$. At the end $B$ believes that he shares the secret $s$ with $A$ while he actually shares $s$ with $C$.

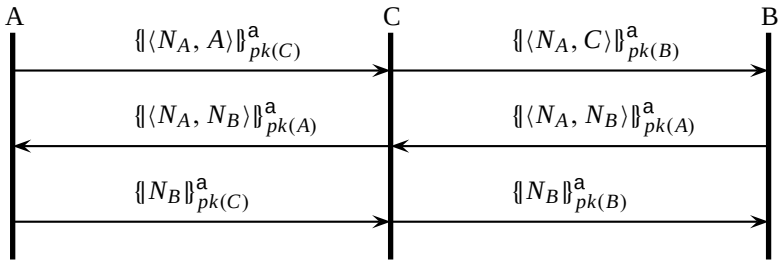
The protocol can be easily corrected by adding the identities of the intended participants as depicted in Figure 1(c).

### *1.2. Needham-Schroeder public key protocol*

A famous example of a protocol is the Needham-Schroeder public key (NSPK) protocol. It was one of the first protocols that was discovered to be flawed using formal methods [Low96a]. The protocol is depicted in Figure 2(a). The protocol aims at achieving mutual authentication while only using asymmetric encryption. $N_A$ and $N_B$ represent nonces, i.e. random numbers freshly generated by $A$, respectively $B$. The rationale of the protocol is that when $A$ receives the second message it must originate from $B$ as it contains the fresh nonce $N_A$ which could only be extracted by $B$ from the first message (due to the encryption of the first message with $B$'s public key). Similarly, freshness of the nonce $N_B$ should convince $B$ that the third message originates from $A$.




(a) NSPK protocol




(b) Man in the middle attack

**Figure 2.** Needham Schroeder Public Key protocol

However, similarly to the handshake protocol it is possible to mount a man in the middle attack if $A$ initiates a session with a dishonest participant $C$. In that case $C$ can successfully authenticate to $B$ masquerading as $A$. Moreover $C$ learns the *a priori* secret nonces $N_A$ and $N_B$. The attack is described in Figure 2(b). Simply adding the sender identity in the second message, i.e. resulting in the message $\{\!|\langle N_A, \langle N_B, B\rangle\rangle|\!\}^{\mathsf{a}}_{pk(A)}$, fixes the protocol.

## 2. Formal models

One of the main advantages of formal methods is to provide a clear, well-defined mathematical model that allows to reason about the capacity of an attacker and to precisely state the security guarantees achieved by a protocol in the given model. A large variety of models have been proposed so far, proposing different trade-offs between expressivity and the possibility to automate proofs.

While there exist a great variety of formal models, all of them have in common the use of a term algebra for describing the messages exchanged during protocol execution. Intuitively, a term algebra allows to model the structure of the messages, abstracting away from the underlying cryptographic details of each primitive.

### *2.1. Term algebra*

A term algebra is built over a set of variables and a *signature*, that is, a set of function symbols given with their arity. A typical signature is $\mathcal{F} = \{\{\!|\_|\!\}^{\mathsf{s}}_{\_}, \langle \_, \_\rangle\}$ where the func-

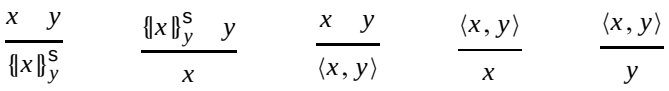


**Figure 3.** Dolev-Yao deduction system.

tion symbols are of arity 2 and model symmetric encryption and concatenation respectively.

The intruder capacities are then often represented using a *deduction system*. The classical deduction system (often referred to as the Dolev-Yao system) for concatenation and (symmetric) encryption is given by the five rules of Figure 3.

The exact term algebra varies from one model to the other.

1. Some primitives may be added such as asymmetric encryption (denoted by $\{\!|\_|\!\}^{\mathsf{a}}_{\_}$) signatures (denoted by $[\_]\_$) or hash functions (denoted by $\mathsf{h}(\_)$).
2. To reflect the *probabilistic* nature of encryption, a third argument may be added when modelling encryption: the same message $m$ encrypted at two distinct time with the same key $k$ does not yield the same cipher-text. The encryption of $m$ by $k$ is then modeled by the term $\{\!|m|\!\}^{\mathsf{s}}_{k,r}$ where $r$ represents the randomness used by the encryption algorithm.
3. A last important difference that is introduced in some models is the use of *explicit destructors*. Most often, the ability to encrypt and decrypt messages is modeled by a deduction system like the one presented in Figure 3. An alternative approach consists in explicitly introducing a functional symbol $\mathsf{dec}$ for decryption together with the equation

   $$\mathsf{dec}(\{\!|x|\!\}^{\mathsf{s}}_{y}, y) = x.$$

   These two ways of modeling encryption are similar but not equivalent. For example, Millen [Mil03] has shown that some attacks can be detected only when destructors are explicitly represented. One of the advantages of using equational theories is to reflect in a natural way the properties of the underlying primitives. Indeed destructors correspond to functions that are actually available to an attacker. Many complex cryptographic primitives such as blind signatures, re-encryption [DKR09], Exclusive Or, Diffie-Hellman exponentiation and non-interactive zero-knowledge proofs [BHM08b] have been modeled by these means.

Different term algebra will be discussed throughout the chapters of this book.

### *2.2. A variety of formal models*

Several symbolic models have been proposed for cryptographic protocols. A unified model would enable better comparisons between each result but such a unified model does not exist currently. The reason for having several popular symbolic models probably comes from the fact that symbolic models have to achieve two antagonistic goals. On the one hand, models have to be as fine grained and expressive as possible in order to better reflect protocol behaviors. One the other hand, models have to remain relatively simple in order to allow the design of (automatic) decision procedures.

Without aiming at an exhaustive list we mention below several symbolic models in order to illustrate the kind of techniques and frameworks that have been used for security protocol verification. In this book most of these models will be described in more details.

*Early models.* One of the first symbolic models dedicated to security protocols has been developed by Dolev *et al.* [DY81,DEK83]. Protocols are described by rewrite rules on words (describing a sequence of encryptions and decryptions). A word $s$ is secret if it is not reachable by rewriting. Merritt *et al.* [DLM82,Mer83] have developed during the same period of time a model where messages are also represented by words. These models are however not expressive enough to reflect primitives such as concatenation or key generation. More complex and detailed models have then been proposed. For example, Paulson [Pau98] has developed a transition-based model where each emission of a message corresponds to an event. Protocol rules then specify possible transitions between sets of events. Similarly, Meadows *et al* [Mea96] have proposed a language (NPATRL) for specifying protocols, also based on events.

*Rewrite rules.* Several models represent the protocol itself as well as the intruder capabilities by rewrite rules. The main models are the multiset rewriting (MSR) model by Mitchell *et al.* [CDL$^+$99,BCJS02], the model based on rewrite rules by Rusinowitch and Turuani used in the Casrul tool [RT01] and Compton and Dexter's model [CD99] based on linear logic.

*Horn clauses.* A variation of the modeling using rewrite rules is the modeling of protocols and attacker actions using Horn clauses [Wei99,Bla01,Bla04,VSS05,CLC03, Gou08]. The modeling in terms of Horn clauses allows to reuse existing results such as different resolution strategies. One of the most successful tools for verifying an unbounded number of sessions is the ProVerif tool developed by Blanchet [Bla01,Bla05, BAF08] which implements a specialised resolution strategy. A detailed description of this approach and Blanchet's algorithm will be given in chapter *"Using Horn Clauses for Analyzing Security Protocols"*.

*Strand spaces.* The strand space model [THG99,GT01] is a special purpose model for reasoning about the traces generated by security protocols. One appealing feature of the model is that it has an intuitive graphical representation of the protocol executions. Moreover, Guttman obtained several composition results in this model [GT00,Gut04, Gut09].

*Constraint systems.* Constraint systems as a symbolic representation of the execution of a bounded number of sessions were first introduced by Millen and Shmatikov [MS01, MS03] and later also developed by Comon-Lundh [CL04]. It is in particular the underlying model in which NP-completeness of secrecy has been proved by Rusinowitch and Turuani [RT01], for a bounded number of sessions. This result will be presented in the chapter *"Verifying a bounded number of sessions and its complexity"*. A more general presentation of constraint systems and a decision procedure will be given in the chapter *"Constraint solving techniques and enriching the model with equational theories"*.

*Process algebras.* A natural modelling of protocols which is closer to an actual implementation is in terms of process algebras. Each role of a protocol corresponds to an independent process. Process algebras (such as CSP [Sch97], the CCS variant CryptoSPA [FM99], the spi-calculus [AG97] or the applied pi calculus [AF01]) provide com-

munication primitives for sending and receiving messages, restriction, parallel composition and replication of processes. This yields an accurate (symbolic) modelling of protocols, with in particular generation of fresh nonces (using restriction) and the possibility for an unbounded number of sessions by using replication.

The main difference between the different models based on process algebra lies in the definition of security properties. Many models [AL00,ALV02,Sch96,Sch97,Bor01] use reachability properties of the form: $P \xrightarrow{*} err$. Other models [AG98,BDNP99,AG97, AF01] base their property definition on an observational equivalence allowing to model a wider range of properties including for instance anonymity and coercion-resistance in voting protocols [KR05,DKR09].

*Logics for security protocols.* Another kind of reasoning about security protocols is to use a Floyd-Hoare style logic. As for program analysis, assertions about the protocol are propagated according to a set of rules. This type of logics goes back to the famous BAN logic [BAN89] and has known many extensions and variations in the early nineties. Recently, a new effort in this direction has been made by Mitchell's group at Stanford with a particular emphasis on compositionality resulting in the Protocol Composition logic which will be described in the corresponding chapter of this volume.

### *2.3. Security properties*

Cryptographic protocols aim at ensuring various security goals, depending on the application. The two most classical security properties are secrecy and authentication. Most of the verification techniques have been developed for these two properties.

*Secrecy* Secrecy is one of the most standard properties: a protocol ensures the confidentiality of some data $s$ if this data is only known to participants which are entitled to access the data. It is usually specified using a reachablity-based property: a protocol is said to preserve the confidentiality of some data $s$ if no execution yields a state such that an attacker is able to learn $s$. In the context of process algebras like the applied pi-calculus [AF01], it is possible to specify a stronger property: a data $s$ is said secret if an attacker cannot distinguish a session of the protocol where $s$ has been used from a session where $s$ has been replaced by an arbitrary data $s'$. This property is often referred to as *strong secrecy*.

*Authentication* An authentication protocol should typically enable an agent to prove her identity. Authentication properties are typically specified by requiring that for any execution of a protocol where an agent $B$ believes that he has received a message $m$ from $A$, then $m$ has been indeed sent by $A$. Many variants have been proposed for authentication, e.g. by Schneider [Sch97] and by Lowe [Low97].

*Equivalence-based properties* While secrecy and authentication goals are usually specified as reachability properties, more complex properties such as privacy-like properties usually require the use of equivalence-based definitions. Strong secrecy, described above, is a first example of an equivalence-based definition. Equivalence-based properties are even more crucial when specifying anonymity-like properties. For example, a protocol ensures anonymous communication if an agent cannot linked the received messages to their respective senders. Several formal definitions of anonymity have been proposed [Aba02,SH02,Low02]. In particular, the definition proposed by Shmatikov

and Hughes relies on observational equivalence. Other examples of security properties stated as equivalence-based properties are privacy in e-voting protocols, receipt-freeness or coercion-resistance [DKR09]. Equivalence-based properties can also be used for analysing security properties specified in cryptographic models [CLC08].

## 3. Outline of the book

Formal methods for analyzing security protocols have reached a good level of maturity. Many algorithms and tools have been proposed and have been successfully applied to a wide range of protocols. The goal of this book is twofolds. First, it presents several foundational techniques for security protocol verification that have given raise to many extensions or which are the key part of successful analysis tools. Second, it presents several well known symbolic models for security protocols, showing the advantages of each of them.

Even relatively simple properties such as secrecy and authentication are undecidable for security protocols [DLMS99]. Hence, algorithms for automated analysis of security protocols either consider restrictions on the protocols or the proposed techniques are incomplete. Both kind of approaches will be covered in this book.

A first mean to recover decidability is to consider a bounded number of sessions, that is, to consider a limited (fixed in advance) number of executions of the protocol. The chapter *"Verifying a bounded number of sessions and its complexity"* presents one of the first decidability and complexity result for analysing a protocol for a bounded number of sessions. It is based on constraint systems which have then been intensively reused in the field of security protocol verification. It has also given birth to an automated tool [ABB+02,ABB+05]. While many early models considered free term algebras it is now widely recognized that this is not sufficient for several important cryptographic primitives. For example, exclusive or is a frequently used operator in security protocols. It admits several algebraic properties, in particular associativity and commutativity. The chapter *"Constraint solving techniques and enriching the model with equational theories"* extends the previous chapter in order to consider equational theories, in particular for associative and commutative operators. The chapter *"Analysing Security Protocols using CSP"* presents a technique for analysing protocols specified using a process algebra. In particular this technique allowed the discovery of the famous man-in-the-middle attack of the Needham-Schroeder public key protocol [Low96a].

To analyze protocols without bounding the number of sessions, several incomplete techniques have been developed which have been shown very successful in practice. The chapter *"Using Horn clauses for analyzing protocols"* presents an algorithm for analysing protocols modeled in Horn clauses. This algorithm is in the heart of the very successful tool ProVerif [Bla05]. The chapter *"Applied pi calculus"* presents the applied pi-calculus, a process algebra which allows to specify complex protocols such as e-voting protocols. The chapter *"Types for security protocols"* then proposes a verification technique for a cryptographic pi-calculus, based on a type systems. Mitchell *et al* have developed a logic (PCL) that allows to analyse protocols in a modular way. This logic is presented in the chapter *"Protocol Composition Logic"*. Guttman *et al.* have developed the *strand spaces* model. This model allows to directly reason on the graphical representation of protocol executions. The model and some of its associated verification

techniques are presented in the chapter *"Shapes: surveying crypto protocols runs"*. Another possibility for analysing protocols is to compute on over-approximation of the attacker behavior, still showing the security of the protocol. This is the intuitive goal of the *rank functions* defined by Schneider and presented in the chapter *"Security analysis using rank functions in CSP"*.

Formal models differ significantly from computational ones. In modern cryptography, security definitions are based on complexity theory. Messages are modeled by bit-strings and encryption functions are algorithms on bit-strings. The issue is then to detect whether an adversary (a Turing machine) is able to learn a confidential information in reasonable (polynomial) time with non negligible probability. This notion of security seems to better reflect the class of attacks that can be mounted in practice. However, security proof are error-prone and difficult to automate. Computational and symbolic models have been developed separately since the 80s. They seem *a priori* very distinct and difficult to conciliate. However, a recent line of research has developed a bridge between the two approaches. In particular, Abadi and Rogaway [AR00] have shown that the cryptographic indistinguishability of sequences of messages can be abstracted by the symbolic equivalence of the corresponding sequences of terms. This result has then been followed by many extensions. To conclude this book, the chapter *"Computational soundness: the case of Diffie-Helman keys"* illustrates this new line of research by presenting a soundness result between symbolic and cryptographic equivalences for Diffie-Hellman encryption.

## References


[Aba02] M. Abadi. Private authentication. Proc. of Workshop on Privacy Enhancing Technologies, 2002.

[ABB+02] A. Armando, D. A. Basin, M. Bouallagui, Y. Chevalier, L. Compagna, S. Bödersheim, M. Rusinowitch, M. Turuani, L. Viganò, and L. Vigneron. The AVISS security protocol analysis tool. In Ed Brinksma and Kim Guldstrand Larsen, editors, *Proc. of the 14th International Conference of Computer Aided Verification (CAV 2002)*, volume 2404 of *LNCS*, pages 349–353. Springer, July 2002.

[ABB+05] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the automated validation of internet security protocols and applications. In K. Etessami and S. Rajamani, editors, *17th International Conference on Computer Aided Verification, CAV'2005*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285, Edinburgh, Scotland, 2005. Springer.

[ACC+08] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, and Llanos Tobarra Abad. Formal analysis of saml 2.0 web browser single sign-on: Breaking the saml-based single sign-on for google apps. In *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*, pages 1–10, 2008.

[AF01] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, January 2001.

[AG97] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. of the 4th ACM Conference on Computer and Communications Security (CCS'97)*, pages 36–47. ACM Press, 1997.

[AG98] M. Abadi and A. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, 1998.

[AL00] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *Proc. of the 12th International Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *LNCS*, pages 380–394, 2000.

[ALV02] R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, 2002.

[AR00] M. Abadi and P. Rogaway. Reconciling two views of cryptography. In *Proc. of the International Conference on Theoretical Computer Science (IFIP TCS2000)*, pages 3–22, August 2000.

[BAF05] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005)*, pages 331–340. IEEE Computer Society, June 2005.

[BAF08] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.

[BAN89] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Proc. of the Royal Society*, volume 426 of *Series A*, pages 233–271. 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in ACM Transactions on Computer Systems 8, 1 (February 1990), 18-36.

[BCJS02] F. Butler, I. Cervesato, A. Jaggard, and Andre Scedrov. A formal analysis of some properties of kerberos 5 using MSR. In S. Schneider, editor, *Proc. of 15th IEEE Computer Security Foundations Workshop (CSFW'15)*, pages 175–190. IEEE Computer Society Press, June 2002.

[BDNP99] M. Boreale, R. De Nicola, and R. Pugliese. Proof techniques for cryptographic processes. In *Logic in Computer Science*, pages 157–166, 1999.

[BHM08a] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proceedings of 21st IEEE Computer Security Foundations Symposium (CSF)*, June 2008.

[BHM08b] Michael Backes, Catalin Hritcu, and Matteo Maffei. Type-checking zero-knowledge. In *Proceedings of 2008 ACM Conference on Computer and Communication Security (CCS'08*, 2008.

[Bla01] 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.

[Bla04] Bruno Blanchet. Automatic proof of strong secrecy for security protocols. In *IEEE Symposium on Security and Privacy (SP'04)*, pages 86–100, Oakland, California, May 2004.

[Bla05] Bruno 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.

[Bla08] Bruno Blanchet. *Vérification automatique de protocoles cryptographiques : modèle formel et modèle calculatoire. Automatic verification of security protocols: formal model and computational model*. Mémoire d'habilitation à diriger des recherches, Université Paris-Dauphine, November 2008. En français avec publications en anglais en annexe. In French with publications in English in appendix.

[Bor01] M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proc. of the 28th Int. Coll. Automata, Languages, and Programming (ICALP'01)*. Springer Verlag, July 2001.

[CC01] H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. In *Research Report LSV-01-13*, december 2001.

[CD99] K. Compton and S. Dexter. Proof techniques for cryptographic protocols. In *Proc. of the 26th International Colloquium on Automata, Languages, and Programming (ICALP'99)*, July 1999.

[CDL+99] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *Proc. of the 12th IEEE Computer Security Foundations Workshop (CSFW'99)*, pages 55–69. IEEE Computer Society Press, 1999.

[CDL06] V. Cortier, S. Delaune, and P. Lafourcade. A Survey of Algebraic Properties Used in Cryptographic Protocols. *Journal of Computer Security*, 14(1/2006), 2006.

[CKS01] R. Chadha, M.I. Kanovich, and A. Scedrov. Inductive methods and contract-signing protocols. In P. Samarati, editor, *8-th ACM Conference on Computer and Communications Security*, pages 176–185. ACM Press, November 2001.

[CL04] H. Comon-Lundh. Résolution de contraintes et recherche d'attaques pour un nombre borné de sessions. Available at `http://www.lsv.ens-cachan.fr/~comon/CRYPTO/bounded.ps`, 2004.

[CLC03] 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.

[CLC08] Hubert Comon-Lundh and Véronique Cortier. Computational soundness of observational equiv-

alence. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*, Alexandria, Virginia, USA, October 2008. ACM Press.

[Cre08] Cas Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, USA, Proc.*, volume 5123/2008 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008.

[DEK83] D. Dolev, S. Even, and R.M. Karp. On the security of ping-pong protocols. In R.L. Rivest, A. Sherman, and D. Chaum, editors, *Proc. of CRYPTO 82*, pages 177–186. Plenum Press, 1983.

[DKR09] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[DLM82] R. Demillo, N. Lynch, and M. Merritt. Cryptographic protocols. In *Proc. of the $14^{th}$ ACM SIGACT Symposium on Theory of Computing*. ACM, May 1982.

[DLM04] Nancy A. Durgin, Patrick Lincoln, and John C. Mitchell. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[DLMS99] 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.

[DY81] D. Dolev and A.C. Yao. On the security of public key protocols. In *Proc. of the 22nd Symp. on Foundations of ComputerScience*, pages 350–357. IEEE Computer Society Press, 1981.

[FM99] Riccardo Focardi and Fabio Martinelli. A uniform approach for the definition of security properties. In *Proc. of World Congress on Formal Methods (FM'99)*, volume 1708 of *Lecture Notes in Computer Science*, pages 794–813. Springer, 1999.

[Gou08] Jean Goubault-Larrecq. Towards producing formally checkable security proofs, automatically. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 224–238, Pittsburgh, PA, USA, June 2008. IEEE Computer Society Press.

[GT00] Joshua D. Guttman and F. Javier Thayer. Protocol independence through disjoint encryption. In *Proc. 13th Computer Security Foundations Workshop (CSFW'00)*, pages 24–34. IEEE Comp. Soc. Press, 2000.

[GT01] J.D. Guttman and F.J. Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 2001.

[Gut04] Joshua D. Guttman. Authentication tests and disjoint encryption: a design method for security protocols. *Journal of Computer Security*, 12(3–4):409–433, 2004.

[Gut09] Joshua D. Guttman. Cryptographic protocol composition via the authentication tests. In *Foundations of Software Science and Computation Structures (FOSSACS'09)*, Lecture Notes in Computer Science, March 2009.

[KK05] D. Kähler and R. Küsters. Constraint Solving for Contract-Signing Protocols. In M. Abadi and L. de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR 2005)*, volume 3653 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2005.

[KR02] S. Kremer and J.-F. Raskin. Game analysis of abuse-free contract signing. In Steve Schneider, editor, *Proc. of the 15th Computer Security Foundations Workshop (CSFW'02)*, pages 206–220. IEEE Computer Society Press, June 2002.

[KR05] Steve Kremer and Mark. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In Mooly Sagiv, editor, *Programming Languages and Systems –Proceedings of the 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200, Edinburgh, U.K., April 2005. Springer.

[KT09] R. Küsters and T. Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. In *2009 IEEE Symposium on Security and Privacy (S&P 2009)*, pages 251–266. IEEE Computer Society, 2009.

[Low96a] 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.

[Low96b] G. Lowe. Some new attacks upon security protocols. In *Proc. of the 9th Computer Seurity Foundation Workshop (CSFW'96)*, pages 162–169. IEEE Computer Society Press, 1996.

[Low97] G. Lowe. A hierarchy of authentication specification. In *Proc. of the 10th Computer Security Foundations Workshop (CSFW'97)*. IEEE Computer Society Press, 1997.

[Low02] G. Lowe. Analysing Protocols Subject to Guessing Attacks. In *Proc. of the Workshop on Issues in the Theory of Security (WITS '02)*, 2002.

[Mea96] Catherine Meadows. Language generation and verification in the nrl protocol analyzer. In *Proceedings of the 9th Computer Security Foundations Workshop (CSFW'96)*. IEEE Computer Society Press, 1996.

[Mer83] Michael J. Merritt. *Cryptographic Protocols*. PhD thesis, Georgia Institute of Technology, February 1983.

[Mil84] Jonathan K. Millen. The interrogator: A tool for cryptographic protocol security. In *IEEE Symposium on Security and Privacy*, pages 134–141, 1984.

[Mil03] J. Millen. On the freedom of decryption. *Information Processing Letters*, 2003.

[MS01] 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.

[MS03] J. Millen and V. Shmatikov. Symbolic protocol analysis with products and diffie-hellman exponentiation. In *Proc. of the 16th IEE Computer Seurity Foundation Workshop (CSFW'03)*, 2003.

[Pau98] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1):85–128, 1998.

[RT01] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190. IEEE Computer Society Press, 2001.

[Sch96] S. A. Schneider. Security properties and CSP. In *Proc. of the Symposium on Security and Privacy*, pages 174–187. IEEE Computer Society Press, 1996.

[Sch97] S. Schneider. Verifying authentication protocols with CSP. In *Proc. of the 10th Computer Security Foundations Workshop (CSFW'97)*. IEEE Computer Society Press, 1997.

[SH02] V. Shmatikov and D.J.D Hughes. Defining Anonymity and Privacy (extended abstract). In *Proc. of the Workshop on Issues in the Theory of Security (WITS '02)*, 2002.

[THG99] J. Thayer, J. Herzog, and J. Guttman. Strand spaces: proving security protocols correct. *IEEE Journal of Computer Security*, 7:191–230, 1999.

[VSS05] Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. On the complexity of equational horn clauses. In *Proc. of the 22th International Conference on Automated Deduction (CADE 2005)*, Lecture Notes in Computer Science, pages 337–352. Springer-Verlag, 2005.

[Wei99] C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In H. Ganzinger, editor, *Proc. of the 16th International Conference on Automated Deduction, CADE'99*, volume 1632 of *LNCS*, pages 378–382, 1999.