A Survey of Algebraic Properties Used in Cryptographic Protocols*

Véronique Cortier[1] **, Stéphanie Delaune[2,3], and Pascal Lafourcade[3,4]

[1] LORIA, CNRS UMR 7503 & INRIA Lorraine project CASSIS,
Campus Scientifique - BP 239, 54506 Vandoeuvre-les-Nancy cedex, France,
cortier@loria.fr
[2] France Télécom R&D
[3] Laboratoire Spécification et Vérification,
ENS de Cachan & CNRS UMR 8643 & INRIA Futurs project SECSI
61, avenue du Président Wilson, 94235 Cachan cedex, France
{delaune,lafourca}@lsv.ens-cachan.fr
[4] LIF, Université Aix-Marseille 1 & CNRS UMR 6166,
39, r. Joliot-Curie 13453 Marseille Cedex 13, France,

Abstract: Cryptographic protocols are successfully analyzed using formal methods. However, formal approaches usually consider the encryption schemes as black boxes and assume that an adversary cannot learn anything from an encrypted message except if he has the key. Such an assumption is too strong in general since some attacks exploit in a clever way the interaction between protocol rules and properties of cryptographic operators. Moreover, the executability of some protocols relies explicitly on some algebraic properties of cryptographic primitives such as commutative encryption. We give a list of some relevant algebraic properties of cryptographic operators, and for each of them, we provide examples of protocols or attacks using these properties. We also give an overview of the existing methods in formal approaches for analyzing cryptographic protocols.

Keywords: cryptographic protocols, algebraic properties, decidability results, exclusive or, Abelian group, modular exponentiation, elliptic curves.

* This work has been partially supported by the RNTL project PROUVÉ-03V358 and 03V360 and the ACI-SI Rossignol.
** Corresponding author: Véronique Cortier, LORIA, CNRS & INRIA Lorraine project CASSIS, Campus Scientifique - BP 239, 54506 Vandoeuvre-les-Nancy cedex, France, *tel:* +33 3 83 59 30 55, *fax:* +33 3 83 41 30 79, cortier@loria.fr

Table of Contents

1 Introduction

Cryptographic protocols are short programs designed to ensure secure communications on channels that may be controlled by an attacker. Considering the increasing size of networks and their dependence on cryptographic protocols, a high level of assurance is needed in the correctness of such protocols. These protocols are notoriously difficult to design and test, and serious flaws have been found in many protocols. Consequently, there has been a growing interest in applying formal methods for validating cryptographic protocols. These protocols use cryptographic primitives such as public and symmetric encryption. These functions are based on mathematical notions (like modular exponentiation or elliptic curves) and on algorithmically hard problems (such as extracting the modular logarithm or factorization into prime numbers). A first approach is to verify a protocol with its actual cryptographic primitives and to show that attacking the protocol can be reduced to solving an algorithmically hard problem. Such proofs are done by hand and are often long and difficult. In particular, they seem very hard to automate.

Another approach is to abstract from cryptographic primitives. This may be justified by the observation that many attacks rely only on the logical structure of the protocols and simply consist of replaying some messages at the right steps. That is why formal methods usually consider encryption schemes as black boxes and assume that an adversary cannot learn anything from an encrypted message except if he has the key. This is called the *perfect encryption assumption*. More generally, formal methods assume *perfect cryptography*: the other cryptographic primitives (like pairing or hashing) are also idealized in order to enable automatic verification. Even if these assumptions are not realistic, many real attacks have been discovered using this approach. The most famous flaw is the man-in-the-middle attack on the Needham-Schroeder protocol with public key encryption, found by G. Lowe [64] using an automatic tool.

Many decidability results have been obtained under this perfect cryptography hypothesis: the secrecy preservation is co-NP-complete for a bounded number of sessions [95], and decidable for an unbounded number of sessions under some additional restrictions [66, 42, 28]. Many tools have also been developed to automatically verify cryptographic protocols like [72, 65, 10].

Recent works investigate how to relax the perfect cryptography assumption by refining the abstraction on cryptographic primitives. The aim is to take into account some of the algebraic properties of the cryptographic primitives. Most of the algebraic properties studied so far and presented in this survey are properties that can be modeled using equations. For example a commutative encryption is expressed by the equality $\{\{x\}_y\}_z = \{\{x\}_z\}_y$. Such a representation of the algebraic properties is natural for many cryptographic primitives and very convenient since it enables one to reuse classical methods on first order logic for terms modulo equational theories. The interest of studying the algebraic properties of the cryptographic primitives is that some attacks may be missed when abstracting encryption as a perfect black box. For example, Bull's protocol has been proven secure [90] under the perfect cryptography assumption. However; the protocol uses the exclusive or operator, which is associative, commutative, nilpotent, and has a neutral element. An attack relying on these properties has been found on this protocol [96]. Even without searching for attacks, one may need algebraic properties to simply be able to specify some protocols. For example, the simple Three Pass protocol proposed by R. Shamir [24] requires a commutative encryption like the RSA encryption. All these examples will be developed in the following sections.

The aim of this survey is to present some algebraic properties of cryptographic primitives relevant for protocols. Note that these properties may be relevant in two different ways:

– Either an attack on a protocol may rely on some properties of the encryption function. For example a protocol may be or may not be secure depending on which encryption function is used. This is the

case of the public-key Needham-Schroeder-Lowe protocol which is insecure when the encryption function uses the ECB or CBC encryption schemes and secure assuming perfect encryption.
- Or a protocol itself may make use of algebraic properties. In that case, it is impossible to describe such a protocol in a model which does not handle algebraic properties.

This survey contains two main sections. In Section 2, we first present an overview of decidability results or semi-decision procedures that have been obtained so far for some algebraic properties using formal methods. After a brief description on how cryptographic protocols are modeled (Section 2.1), we give in Section 2.2 a summary of the results obtained assuming perfect cryptography. Then in Section 2.3, we provide, for each algebraic property, a list of known results. We summarize these results in two tables. In Section 3, we present a survey of the algebraic properties used by cryptographic protocols. For each of them, we provide examples of protocols or attacks on protocols that make use of the property. The properties we present are mainly those of associativity, commutative encryption, exclusive or, Abelian groups, modular exponentiation, homomorphism and elliptic curves.

2 Existing Results

Many decidability results have been obtained under the perfect cryptography assumption. Recently, several works try to extend these results to protocols with some algebraic properties. We give here an overview of these two kinds of results, after briefly describing how cryptographic protocols are modeled in formal methods. Results under the perfect cryptography assumption are summarized in Table 1, results for algebraic properties are summarized in Table 2.

2.1 Modeling Cryptographic Protocols

Security protocols are typically specified as sets of roles which are abstract patterns of communication specifying which messages are sent, and how to respond to the reception of any message. The messages are represented by terms built over a given alphabet of function symbols containing constants, pairing, and encryption. It may also contain some other symbols such as decryption, exclusive or, and multiplication. In such a case we cannot continue to model messages in terms of free algebras. Instead, we have to consider in addition an equational theory defined by a set of equations to take into account algebraic properties of the operators.

While there are many properties that a cryptographic protocol may aim to guarantee, the main results relaxing the perfect cryptography assumption regard only trace properties (*i.e.* properties on the sequences of messages that may be sent during the execution of the protocol), and in particular *secrecy*: a secret, generated by an honest agent, should not be leaked to the intruder. For verifying such a property the attacker is typically represented as an active entity who is assumed to have a complete control of network communication: he is able for instance to eavesdrop and replay messages, impersonate honest agents, and generate nonces. Deciding whether a protocol preserves secrecy against such an active intruder is called the *security problem*. However, the security decision problem in the presence of a passive attacker, who can only eavesdrop messages, is also a significant question and is in general the first step to obtain decision procedures for the security problem and the search for attacks. We can formulate this *intruder deduction problem*, in the following way: given a finite set of messages T and a message s (the secret), can the intruder deduce s from T? We present existing results for both the security problem and the intruder deduction problem.

Although the results have been obtained in different models (multiset rewriting, strand spaces, process calculus, ...) we give the results without specifying the models since it is quite well accepted that these results are in general also valid in the other models. Some translations between different models have been proposed *e.g.* by [9].

2.2 Results under the Perfect Cryptography Assumption

The analysis techniques discussed in this section and summarized in Table 1 assume perfect cryptography. This means that cryptographic primitives (pairing, encryption, ...) are considered without any algebraic properties. In particular, under the perfect encryption assumption, the encryption is modeled as a black box and the only way to obtain the plain text from the cipher text is by knowing the decryption key. We are going to give a brief overview of decidability results concerning the security problem.

<table>
<tr><th rowspan="2">Bounded number of sessions</th><th colspan="2">Unbounded number of sessions</th></tr>
<tr><th>Without nonces</th><th>With nonces</th></tr>
<tr><td rowspan="4">co-NP-complete
[95]</td><td>General case:
Undecidable
[45, 26]</td><td>General case:
Undecidable
[45, 26, 42, 3]</td></tr>
<tr><td>Bounded message length:
DEXPTIME-complete
[42, 22]</td><td>Bounded message length:
Undecidable [42, 3]
with non-unifiable subterms:
Decidable [93]</td></tr>
<tr><td>Tagged protocols:
EXPTIME
[12]</td><td>Strongly typed protocols
Inferable identities:
Decidable [66]
Tagged protocols:
Decidable [94]</td></tr>
<tr><td>One copy:
DEXPTIME-complete
[28, 99]</td><td>Ping-pong protocols:
PTIME [34]</td></tr>
</table>

Table 1. Summary of Results for the Security Problem under the Perfect Cryptography Assumption.

Some Undecidability Results

Though cryptographic protocols are often described in a concise way, the verification problem is difficult because of many sources of unboundedness in their modeling, for instance the number of sessions, the length of messages, or the nonce generation.

When considering an unbounded number of sessions, the main sources of undecidability are the nonce generation and the possibility to copy arbitrary messages. Several codings of the Post Correspondence Problem [35] have been proposed, *e.g.* S. Even and O. Goldreich show in this way the undecidability of the security problem using only a bounded number of nonces [45]. Some other codings exist in order to obtain subtler undecidability results, for example H. Comon-Lundh and V. Cortier show in [26] that the problem is undecidable even without using composed keys. In both cases, the undecidability results exploit the fact that the size of messages is not bounded.

Using nonce generation, N. Durgin *et al.* [42] show that the security problem for protocols is undecidable, even when the length of the messages is bounded. To prove their undecidability result, they encode existential Horn clauses using both the encryption and pairing primitives. R. Amadio and W. Charatonik [3] are even more careful in their analysis. Their undecidability result, obtained by encoding 2-counter machines, relies only on the encryption primitive and not on pairing.

Some Decidability Results
We have seen that the prominent sources of undecidability are unbounded message length and unbounded number of nonces. Now, we are going to give some decidability results which can be obtained by setting strong conditions on the protocols. One of the first results is a PTIME complexity result which has been obtained by D. Dolev *et al.* for ping-pong protocols between two participants [34]: in each step of the protocol, one of the agents applies a sequence of operators to the last received message, and sends it to the other agent.

In [66], G. Lowe studies the security problem with an unbounded number of nonces and shows decidability for a subclass of protocols. He assumes in particular that each participant can completely analyze any messages he receives, that messages contain no long-term secrets and, that identities are inferable from messages. R. Ramanujam and S. Suresh show that secrecy is decidable in the presence of an unbounded number of nonces, considering two kinds of restrictions. In [93], they show that secrecy is decidable assuming that the message size is bounded and that no encrypted subterm of a message of the protocol can be unified with a subterm of another message. In [94], they obtain a decidability result for both an unbounded number of nonces and an unbounded size of messages, enforcing the non-unifiability condition by assuming that each encrypted subterm is tagged by a fresh nonce. However; in this last case, some protocols such as the Yahalom protocol do not follow the restricted syntax since agents have to forward message components that they cannot decrypt.

On the other side, some works [42, 22, 28] studied the security problem in the setting of a bounded number of nonces and additional strong restrictions on the protocols. For example the technique described in [28] by H. Comon-Lundh and V. Cortier use stringent criteria to show that the security problem is decidable in 3-EXPTIME. They consider protocols in which at each transition an agent may copy at most one unknown component of the received message. H. Seidl and K. Verma [99] studied carefully the complexity of this class of protocols, showing that the security problem is DEXPTIME-complete. Y. Chevalier *et al.* [22] and N. Durgin *et al.* [42] assume that the message size is bounded and obtain that the security problem is DEXPTIME-complete. Some other works such as the work of B. Blanchet [12] use tagging schemes to obtain decidability of secrecy.

Even if it is assumed that there is a bounded number of sessions (thus, also a bounded number of nonces), it is still not easy to design a decision algorithm since the number of messages that can be created by the attacker is unbounded. M. Rusinowitch and M. Turuani extend in [95] the work of R. Amadio *et al.* [4] by giving a co-NP-complete procedure for deciding protocol security for the Dolev-Yao attacker as long as the number of sessions is bounded. Some similar results [13, 76] have been obtained in other models. H. Huttel [53] shows a similar result in a context of process algebra for a stronger secrecy property which says that a datum s is secret if the session which contains this datum is indistinguishable from all the sessions containing a datum s' at the place of s. This notion of secrecy is known as an observational equivalence property.

2.3 Results under Relaxation of the Perfect Cryptography Assumption

Certain algebraic properties of encryption, such as the homomorphic properties of RSA or the properties induced by chaining methods for block ciphers, are widely used in protocol constructions. Many real attacks rely on these properties. Recently, several procedures have been proposed to decide insecurity of cryptographic protocols when considering some algebraic properties of the cryptographic primitives, mostly for a finite number of sessions.

The results presented in this section and summarized in Table 2 are first steps towards reducing the gap between formal methods and mathematical proofs typically employed in cryptographic analysis of security protocols. Even though mathematical properties of the underlying cryptographic primitives are taken into account, the analysis is still done on an abstract model, and thus attacks can be missed due to this idealized treatment of cryptography.

Very few automatic tools can handle algebraic properties. One of the most flexible tool is probably the automatic verifier developed by B. Blanchet [11]. Any property can be defined provided that it can be expressed in Horn clauses. The treatment of equations is still very naïve and preliminary. In particular, the system may not terminate when more complex equations are entered. The NRL protocol analyzer [72] of C. Meadows, based on narrowing techniques, can also handle some equational theories. In the first version, the analyzer could deal with some classical reduction rules. Since it has been enriched [73], it can also analyze group protocols with Diffie-Hellman exponentiation, such as the IKA-1 protocol. In the Casper tool [65], G. Lowe and A.W. Roscoe partially model Vernam encryption (which uses exclusive or) and discover different attacks [68] in the TMN protocol [106]. The Casper tool also enables the analysis of protocols with timestamps under typing assumptions and assumptions on the time window to bound the search space. Finally, the Casrul tool [55] of F. Jacquemard *et al.* is able to find most of the attacks relying on the associativity property of the pairing function.

Theoretical results are much more numerous; we present them in the following sections, sorted by the algebraic properties.

Associativity

Associativity is a common property. This property is modelized by $f(f(x,y),z) = f(x, f(y,z))$, with the usual example of the associativity of the sum $(x+y)+z = x+(y+z)$. To our knowledge, there is no theoretical work studying this single property applied to cryptographic protocols.

However, some tools, such as Casrul [55], partially take the associativity of the pairing function (*i.e.* $[[x,y],z] = [x,[y,z]]$) into account: it manages to find most of the type flaw attacks relying on the associativity of the pairing function. algorithm modulo associativity is not fully In general, they do not capture the full theory of associativity, mostly for reason of efficiency. Such a type flaw attack is presented in Section 3.1.

Commutativity

Chevalier *et al.* present in [20] an NP decision procedure for the security problem of protocols that employ commuting encryption, *i.e.* $\{\{x\}_y\}_z = \{\{x\}_z\}_y$. One of the most important instances of commuting encryption is RSA encryption with common modulus. For ping-pong protocols (described in Section 2.2), M. Turuani [107] shows that the security problem is co-NP-complete for an unbounded number of sessions.

Note that the first result in formal analysis of security protocols that goes beyond the perfect encryption assumption is certainly the one of S. Even *et al.* [46], who are interested in RSA encryption. In particular, they deal with the multiplication operator $\times$ and properties such as the homomorphic properties of RSA, *i.e.* $\{x\}_k \times \{y\}_k = \{x \times y\}_k$. The authors also consider commutative encryption but in a restricted way. Indeed, assuming that different moduli are used to generate different key pairs, they only have to consider commutative encryption between a key and its inverse. This can be encoded by the following cancellation equations $\{\{x\}_k\}_{k^{-1}} = x$ and $\{\{x\}_{k^{-1}}\}_k = x$. They show that

	Intruder deduction problem	**Security problem**	
		Bounded number of sessions	**Unbounded number of sessions**
Commutativity	*PTIME [20]*	*co-NP-complete [20]*	**Ping-pong protocols:** *co-NP-complete [107]*
Exclusive Or	*PTIME [19]* **with homomorphism:** *EXPTIME [62]*	**General case:** *Decidable [29]* **Restricted protocols:** *co-NP-complete [19]*	**One copy - No nonces:** *Decidable [28]* **Two-way automata - No nonces:** *Decidable [108]*
Abelian Groups	*NP [29]* **with homomorphism:** *EXPTIME [62]*	*Decidable [100]*	**Two-way automata - No nonces:** *Decidable [108]*
Homomorphism	*PTIME [30]* **with *AC:*** *NP-complete [62]*		
Prefix	*PTIME [19]*	*co-NP-complete [19]*	
Abelian Groups and Modular Exponentiation	*PTIME [21]*	**General case:** *Decidable [100]* **Restricted protocols:** *co-NP-complete [21]*	**AC properties of the Modular Exponentiation** **No nonces:** *Semi-Decision Procedure [52]*
Timestamps	*PTIME* (1)	*Decidable [16]*	*Semi-Decision Procedure [38]*

(1) For the intruder deduction problem, there is no notion of time. Thus deciding whether a message with timestamps can be deduced from a set of messages with timestamps corresponds to the intruder deduction problem where timestamps are considered as constants known to the intruder.

Empty boxes mean that, to our knowledge, no result has been obtained so far.

Table 2. Summary of Decidability Results or Semi-Decision Procedures for some Equational Theories.

RSA properties are of no concern to the security of ping-pong protocols: if a ping-pong protocol is secure in the abstract model then its implementation using RSA is also secure.

In [95], M. Rusinowitch and M. Turuani extend the intruder model by adding some capabilities of the intruder: from the messages $\{\{m\}_k\}_k$, he can retrieve the plaintext m. This coding does not allow to capture the whole theory of idempotence represented by the equation $\{\{m\}_k\}_k = m$ since the rules can only be applied at the top of messages. However, such a theory falls into a particular class of equational theories that can be treated by a generic result of S. Delaune and F. Jacquemard [36]. This result is detailed at the end of this section.

Some protocols requiring commutativity to be executable are presented in Section 3.2.

Exclusive Or

The $\oplus$ symbol denotes the binary operation called exclusive or, also denoted by XOR. The properties of XOR are:

$$\begin{aligned} x \oplus (y \oplus z) &= (x \oplus y) \oplus z && \text{(associativity)} \\ x \oplus y &= y \oplus x && \text{(commutativity)} \\ x \oplus 0 &= x && \text{(neutral element)} \\ x \oplus x &= 0 && \text{(nilpotence)} \end{aligned}$$

This operation is used in many protocols and has aroused a lot of interest during the last years. H. Comon-Lundh and V. Shmatikov present in [29] a decision procedure based on constraint solving techniques for solving the security problem of cryptographic protocols employing XOR. In [19], Y. Chevalier *et al.* improve this result by abstracting from intruder rules using *so-called* oracle rules, *i.e.* deduction rules that satisfy some conditions. As an instance of the general framework, they obtain that the security problem for a bounded number of sessions is in co-NP for a large class of protocols in the case of an intruder who can exploit the properties of the XOR operator.

In [28], H. Comon-Lundh and V. Cortier prove some decidability results of an extension of the Skolem class of first-order logic for the equational theory of XOR. As an application, they get a decidability result in formal analysis of security protocols in the presence of an unbounded number of sessions. They assume a finite number of nonces (which is a sound abstraction) and suppose that at each transition an agent may copy at most one unknown component of the received message. For another subclass of first-order logic, corresponding to two-way tree automata with Exclusive or, K. Verma [108] also gives a decidability result. He does not attempt to precisely identify the class of protocols that can be handled. Instead, when the protocol rules are not exactly translatable to alternating two-way automata, he does a sound approximation.

The Casper tool [65] considers exclusive or in the case of Vernam encryption. The Vernam encryption of m by m' is simply $m \oplus m'$. G. Lowe models it by adding new deduction rules to the intruder. For example, the intruder is able to get $m \oplus m'$ from m and m', to get m' from $m \oplus m'$ and m, and to get $m \oplus m'$ from $m' \oplus m$. Using this tool, G. Lowe and A.W. Roscoe [68] discover or retrieve some flaws in the TMN protocol [106] and prove the security of an improved version.

Some protocols using this primitive are described in Section 3.3. Those for which exclusive or is used in combination with other operators can be found in Sections 3.5 and 3.10.

Abelian Groups

The $\times$ symbol denotes the multiplicative binary operation of Abelian groups. The properties of Abelian groups are:

$$\begin{aligned} x \times (y \times z) &= (x \times y) \times z && \text{(associativity)} \\ x \times y &= y \times x && \text{(commutativity)} \\ x \times 1 &= x && \text{(neutral element)} \\ x \times x^{-1} &= 1 && \text{(inverse)} \end{aligned}$$

The intruder deduction problem can be decided in non-deterministic polynomial time in the case of Abelian groups. This result has been shown by H. Comon-Lundh and V. Shmatikov in [29].

The security problem has been investigated by J. Millen and V. Shmatikov in [77, 100]. They present in [77] a constraint solving technique that reduces the problem to a system of quadratic Diophantine equations, but decidability of such equations remains an open question. However in [100], V. Shmatikov succeeds in reducing the initial problem to the solvability of a particular system of quadratic Diophantine equations, in this way proving the decidability of the insecurity problem for a bounded number of sessions.

For an unbounded number of sessions and a finite number of nonces, K. Verma [108] gets a decidability result for a restricted class of protocols, corresponding to protocols which can be encoded using two-way tree automata. He extends this work to two other equational theories: the first one defined by the three first equations (associativity, commutativity and neutral element) and the second one defined by the same three first equations plus the equations $-(x+y) = (-x)+(-y)$ and $-(-x) = x$.

An example of a protocol exploiting properties of Abelian groups is described in Section 3.4.

Homomorphism

We consider operators that satisfy equalities of the form $f(g(x,y)) = g(f(x), f(y))$. In [30], H. Comon-Lundh and R. Treinen investigate for which class of equational axioms the standard intruder model can be extended such that the intruder deduction problem is decidable. As an instance of this general framework, they obtain that the intruder deduction problem is decidable in PTIME, in presence of the following homomorphism property:

$$\{[u,v]\}_k = [\{u\}_k, \{v\}_k]$$

This property is in particular satisfied when using block ciphers as Electronic Code Book (ECB). The ECB mechanism is the most obvious way to extend a block cipher to a text of arbitrary length. A block cipher encrypts plain text in fixed-sized-n-bits blocks (often n=64). For messages exceeding n bits, ECB consists in partitioning the message into n-bits blocks and encrypting each of them separately. This property can be used to mount an attack on the Needham-Schroeder-Lowe protocol (see Section 3.5).

In [62], P. Lafourcade *et al.* investigate the intruder deduction problem in presence of several variants of *AC*-like theories in combination with the law of homomorphism:

$$h(x+y) = h(x)+h(y)$$

They obtain an EXPTIME decision procedure when the *AC*-like theory considered for $+$ is the exclusive or theory or the Abelian groups theory. They also show that the problem is NP-complete when the theory considered for $+$ is only *AC*. They consider also the case of homomorphism over a plus symbol defined by $\{x+y\}_k = \{x\}_k + \{y\}_k$. This case is an extension of the previous case where h is the encryption function. The same complexity results are available for the *AC*-like theories under consideration. Such theories are used in a lot of protocols. See for instance, the TMN protocol described in Section 3.5.

Prefix Property

The prefix property is the ability of an intruder to get from an encrypted message the encryption of any of its prefixes: from a message $\{x,y\}_z$, he can deduce the message $\{x\}_z$. This property strongly depends on the encryption algorithm. For example, the ECB algorithm, presented in the previous paragraph, has this property. However; the ECB algorithm is not commonly used. A relatively good method of encrypting several blocks of data is Cipher Block Chaining (CBC). In such a system, the encryption of message block sequence $P_1P_2\cdots P_n$ (where some bits may be added to P_n such that every block has the same length) with the key K is $C_0C_1C_2\cdots C_n$ where $C_0 = I$ (initialization block) and $C_i = \{C_{i-1} \oplus P_i\}_K$. The CBC encryption system has the following property: if $C_0C_1C_2\cdots C_iC_{i+1}\cdots C_n = \{P_1P_2\cdots P_iP_{i+1}\cdots P_n\}_K$ then $C_0C_1C_2\cdots C_i = \{P_1P_2\cdots P_i\}_K$, that is to say an intruder can get $\{x\}_z$ from $\{x,y\}_z$ if the length of x is a multiple of the block length used by the cryptographic algorithm. This property can be used to mount attacks on several well-known protocols. Some of them are described in Section 3.6.

In [60], S. Kremer and M. Ryan notice that one can also reuse any postfix $C_{i+1}\cdots C_n$ of a cipher $C_0C_1C_2\cdots C_n$ as a valid cipher. All we need to do is to set the initialization block I to C_i and we obtain a valid cipher $C_{i+1}\cdots C_n$ corresponding to the encryption of the $n-i$ last plaintext blocks. Most often such attacks can be prevented by including additional integrity mechanism.

The framework developed in [19] by Y. Chevalier *et al.* to study the XOR theory can also be applied to model an intruder that may exploit the prefix property. They show that in this case the security problem remains co-NP-complete.

Abelian Groups and Modular Exponentiation

The $\times$ symbol denotes the multiplicative binary operation of Abelian groups. The properties of the Abelian groups and modular exponentiation theory are those of Abelian groups extended with:

$$\begin{aligned} exp(exp(x,y),z) &= exp(x, y \times z) \\ exp(x,1) &= x \end{aligned}$$

This theory takes into account simple properties of product and exponentiation operators, such as those of RSA and Diffie-Hellman exponentiation, which are widely used in protocol constructions. All the results described below assume that the Abelian groups operator (multiplication) appears only in the exponents. In particular, exponentials are not multiplied with each other. This restriction is necessary to obtain decidability results. Indeed, D. Kapur *et al.* [58] has shown the undecidability of unification modulo the theory of exponentiation when the distributivity property of exponentiation over multiplication is assumed. Now, if unification is undecidable, the security problem is undecidable too. Indeed, in such a case the unification problem of two terms $u[x_1,\dots,x_n]$ and $v[x_1,\dots,x_n]$ with variables $x_1,\dots,x_n$ can be reduced to the security problem for one session of the following protocol when requiring the secrecy of s. The protocol involves 2 principals A and B:

$A: \quad \rightarrow M_1,\dots,M_n$	(A sends a sequence of n messages.)
$B: x_1,\dots,x_n \rightarrow \{u[x_1,\dots,x_n], v[x_1,\dots,x_n]\}_k$	(B answers by instantiating the variables $x_1,\dots,x_n$ by the messages sent by A.)
$A: \quad \{x,x\}_k \rightarrow s$	(k is a fresh key.)

J. Millen and V. Shmatikov mentioned in [77] that the security problem is still an open problem even in case of a fixed generator of the group. Partial results for protocol analysis have been obtained by M. Boreale and M.G. Buscemi [14] and Y. Chevalier *et al.* in [21]. The decision procedure of [14] requires an *a priori* upper bound on the number of factors in each product, and they do not provide a complexity result. Y. Chevalier *et al.* [21] prove that the security problem is co-NP-complete in the presence of Abelian groups and modular exponentiation from arbitrary bases, but under some restrictions on how agents and intruder learn information in products of exponents.

In [100] the security problem is reduced to the solvability of a special decidable system of quadratic equations in the domain of integers, providing the expected decidability result without the restrictions described previously.

J. Goubault-Larrecq *et al.* [52] have developed a practical implementation to verify cryptographic protocols using modular exponentiation on a fixed generator g. This operator is modeled by adding a free symbol *exp* and an associative and commutative operator $\times$. The term $exp(M_1 \times M_2)$ represents the exponentiation $(g^{M_1})^{M_2}$. To represent the ability for an attacker to raise $exp(M_1)$ to the M_2-th power, a deduction rule is added: knowing $exp(M_1)$ and M_2, any agent or attacker can deduce $exp(M_1 \times M_2)$. Using these abstractions, they verify the IKA.1 group-key agreement protocol [6] for up to 4 principals. Note that they do not consider the full algebraic theory of modular exponentiation thus they miss some attacks. In particular, they do not model inverses, which can be used to stage attacks such those described by Pereira and Quisquater [92].

C. Meadows has also extended her NRL protocol analyzer [72] in order to verify group key protocols using Diffie-Hellman exponentiation. She has in particular analyzed the AGDH-2 protocol [73], very similar to the IKA.1 protocol which is described in Section 3.7.

Timestamps

Timestamps are often used in cryptographic protocols to prevent replay of messages communicated in the past. However, in most of the existing verification methods and decidability results for cryptographic protocols, timestamps are replaced by nonces because of the complexity of the verification of time-dependent protocols. As a consequence, temporal properties of timestamps are not taken into consideration although they can be used to mount attacks (see for instance the attack on the Wide Mouthed Frog protocol described in Section 3.11). Most of the works on timed cryptographic protocols uses theorem-provers [44] or finite-state model-checkers [65]. While the first ones need human help, the second ones rely on typing assumptions and assumption on the time window to bound the search space.

More recently, some automatic procedures for proving secrecy have been proposed [38, 16] to deal with time-dependent cryptographic protocols. G. Delzanno and P. Ganty's approach [38] is based on data structures for symbolically representing sets of configurations of an arbitrary number of parallel protocol sessions. Since verification of secrecy for unbounded protocols is undecidable, termination of state exploration cannot be guaranteed in general. In [16], L. Bozga *et al.* present a model inspired by timed automata and a symbolic decision procedure to deal with bounded time-dependent cryptographic protocols. Their approach provides an algorithm (and hence a decidability result) for checking security properties of timed cryptographic protocols for a bounded number of sessions.

Classes of Equational Theories

We have seen that there exist a lot of decision results when considering a fixed equational theory corresponding to a fixed intruder power. However, some papers [36, 2] propose generic decision procedures to solve useful problems for verification of security protocols. These procedures can be applied to any model provided that they can be axiomatized by a convergent rewriting system verifying some syntactic conditions. In [2], M. Abadi and V. Cortier show that the problems of deducibility and indistinguishability (static equivalence) are both decidable in PTIME for convergent subterm theories, *i.e.* theories described by sets of equations $M = N$ where N is a subterm of M. Simultaneously, S. Delaune and F. Jacquemard [36] prove that the security problem (in the presence of an active attacker) is decidable (co-NP-complete) for a class of equational theories which is slightly more restrictive than convergent subterm theories. The class of rewriting systems which are in the scope of these results contains the standard Dolev-Yao theory of [34] and other relevant theories like the theory of idempotence which is mentioned and partially treated in [95]. Moreover the use of explicit destructors such as decryption and projection operators allows us to specify protocols (see [36] for examples) without assuming some kind of integrity checks.

An ongoing work, presented in [27] by H. Comon-Lundh, consists in separating the *offline intruder theory*, *i.e.* the capabilities of the intruder to build new messages, from the equational theory, *i.e.* the properties satisfied by the cryptographic primitives. Observing that several previous works [19, 21] are based on a proof normalization theorem, he aims at stating a proof normalization result, which abstracts the offline intruder theory such as the so-called oracle rules of Y. Chevalier *et al.* in [19], and the equational theory. The aim of H. Comon-Lundh is to encompass previous results obtained for particular equational theories such as exclusive or, Abelian groups, *etc*.

A recent work of J. Millen [75] and another of C. Lynch and C. Meadows [69] compare the approach using explicit destructors to the standard one for the case of the decryption operator. They give conditions under which security for the free algebra implies security for the rewrite rule model. In order to render the analysis done by formal methods closer to more concrete cryptographic models, C. Meadows suggests in [74] a hierarchy of models at varying degrees of abstraction. In such an approach, the choice of the model in which the analysis is performed depends on the conditions verified by the studied protocol.

After this overview on existing results regarding analysis of cryptographic protocols, we give a survey of relevant algebraic properties and for each of them we provide examples of protocols or attacks using these properties.

3 Relevant Algebraic Properties of Cryptographic Primitives

Cryptographic primitives have algebraic properties that come from the use of mathematical functions such as addition, multiplication or modular exponentiation. We present here a list of identified algebraic properties that may be used in the execution of a protocol or for attacking a protocol. We illustrate each property with some examples of protocols or attacks on protocols. While we hope to give a complete overview of algebraic properties, we do not attempt to give a complete list of protocols using these properties but only some examples for the sake of illustration. Note also that the cryptographic primitives used by protocols may exhibit other properties besides the ones described in the corresponding sections. Indeed, for each algebraic property, we provide protocols or attacks on a protocol that really require the property in order to be executable. Actual implementations may satisfy other properties that should also be taken into account in order to safely verify the protocol.

The properties are informally described, using equations, while the protocols are described using the SPORE format [104], which gives a protocol description close to the Clark&Jacob syntax [24]. The notation `I(A)` represents: on the left hand side of an arrow, the intruder `I` masquerading `A` to send a message, and on the right hand side of an arrow, the intruder intercepting a message intended for `A`.

For the sake of clarity and when there is no ambiguity, the concatenation of several terms is written using a sequence `[u1,...,uk]`, rather than nested pairing. In such a case, we can consider that the sequence is parsed from left to right.

3.1 Associativity

Firstly, we consider the associativity theory and we give a protocol against which an attack can be mounted by an active attacker only when the pairing function, denoted by `[_,_]`, is associative, *i.e.* `[[x,y],z] = [x,[y,z]]`.

NEEDHAM-SCHROEDER-LOWE MODIFIED PROTOCOL

Author(s): G. Lowe (1995)
Summary: This protocol is an amended version of the well-known Needham-Schroeder-Lowe public-key authentication protocol. Indeed, the first message in the original version is `{Na,A}PK(B)`, with the nonce first in the encrypted field. This apparently inconsequential difference leads to an attack, as first mentioned in [76]. Here, we assume that each agent initially knows the other's public key.

Protocol specification.

```
A,B:     principal
Na,Nb:   fresh number
PK, SK:  principal -> key (key pair)
   1.  A -> B :  {A,Na}PK(B)
   2.  B -> A :  {[Na,Nb],B}PK(A)
   3.  A -> B :  {Nb}PK(B)
```

Attacks. The attack described below allows an intruder `I` to impersonate another agent `A` to set up a bogus session with `B`. The attack involves two simultaneous runs of the protocol: in run `i`, `I` impersonates `A` to establish a bogus session with `B`; in run `ii`, `I` establishes a valid session with `A` who plays the role of `B`. It relies on the fact that `[[I, Nb], B]=[I, [Nb, B]]`.

```
i.1.    I(A) ->  B   :  {A,I}PK(B)
i.2.      B  -> I(A) :  {[I,Nb],B}PK(A)
ii.1.     I  ->  A   :  {I,[Nb,B]}PK(A)
ii.2.     A  ->  I   :  {[[Nb,B],Na'],A}PK(I)
i.3.    I(A) ->  B   :  {Nb}PK(B)
```

Remark. The attack requires two somewhat implausible type confusions: `I` in the first message is occupying a nonce field, and `[Nb, B]` in the first message of the run `ii` is also occupying a nonce field.

3.2 Commutativity

A function symbol `f` is said to be commutative if `f(x,y) = f(y,x)`. We present here two protocols: the Shamir Three Pass protocol and the Diffie-Hellman protocol, which both require commutativity in order to be executable.

Shamir-Rivest-Adleman Three Pass Protocol

Author(s): A. Shamir, R. Rivest, L. Adleman, *unpublished*
Summary: The following protocol, described in [24], allows two principals to exchange a secret message without sharing any initial secret.

The initiator `A` encrypts his message `M` by his secret key `Ka`, then `B` encrypts the message he received by his secret key `Kb`. Since `{{M}Ka}Kb` = `{{M}Kb}Ka`, the agent `A` can decrypt it and send `{M}Kb` to `B`. Then, using `Kb`, `B` can retrieve `M`.

Protocol specification.

```
A, B :   principal
Ka, Kb : symkey
M :      fresh number
   1.  A -> B :  {M}Ka
   2.  B -> A :  {{M}Ka}Kb
   3.  A -> B :  {M}Kb
```

Requirements This protocol assumes that encryption is commutative, *i.e.* `{{x}y}z` = `{{x}z}y`.

Attacks. A variety of attacks have been found on this protocol [24]. Here we present an attack relying on the fact that the participants are not authenticated.

```
1.     A  -> I(B) :  {M}Ka
2.   I(B) ->  A   :  {{M}Ka}Ki
3.     A  -> I(B) :  {M}Ki
```

Now, the intruder can compute the message `M`.

Diffie-Hellman Key Exchange Protocol

Author(s): W. Diffie and M. Hellman (1978)
Summary: We describe the Diffie-Hellman key exchange algorithm [40].

The initiator A first chooses a prime number P and a primitive root G of the group $\mathbb{Z}/P\mathbb{Z}$. He sends the exponentiation of G by a fresh number Na and the responder does the same with a fresh number Nb. At the end, they share a common key which is the exponentiation of G by Na and Nb. The protocol has to guarantee the secrecy of the fresh key. B. Blanchet proved the secrecy property in [10], considering encryption functions as black boxes. The authenticity is not guaranteed by the protocol, hence the intruder may impersonate A (or B).

Protocol specification.

```
A, B :           principal
P, G, Na, Nb :   fresh number
kap :            number, number, number -> number
   1.  A -> B :  P, G
   2.  A -> B :  exp(G, Na) mod P
   3.  B -> A :  exp(G, Nb) mod P
```

Requirements The protocol assumes that modular exponentiation is commutative, *i.e.*

```
exp(exp(G, x), y) mod P = exp(exp(G, y), x) mod P
```

3.3 Exclusive Or

The $\oplus$ symbol denotes the binary operation called exclusive or. The properties of exclusive or are:

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z \quad \text{(E1)}$$
$$x \oplus y = y \oplus x \quad \text{(E2)}$$
$$x \oplus 0 = x \quad \text{(E3)}$$
$$x \oplus x = 0 \quad \text{(E4)}$$

This operation is used in many protocols, we describe three of them in this section: the Bull protocol, the WEP protocol and the Gong mutual authentication protocol. Some other protocols like the TMN protocol (Section 3.5) and the DES encryption algorithm (Section 3.10) are presented later since exclusive or is used together with other operators. Let us also mention the PAK-Z protocol [61, 71] (not presented in this survey), which uses both exclusive or and modular exponentiation. This protocol based on RSA aims at achieving password authenticated key exchange.

BULL'S AUTHENTICATION PROTOCOL

Author(s): J. Bull (1997)
Summary: This protocol, described in [17], aims at establishing fresh session keys between a fixed number of participants and a server: one key for each pair of agents adjacent in the chain. For example, assume that the protocol is initiated by A and then goes through B and C before reaching S. At the end, new session keys Kab and Kbc are established. Each key Kxy should be known to exactly x and y (and also S), even if some nodes other than x and y are malicious. For the sake of simplicity, we describe the 3-party version of this protocol.

Protocol specification.

```
A, B, C, S :    principal
Kab, Kbc :      fresh symkey
Na, Nb, Nc :    fresh number
Kas, Kbs, Kcs : symkey
h :             message, symkey -> message
   A computes Xa = h([A,B,Na],Kas), [A,B,Na]
   1.  A -> B :  Xa
   B computes Xb = h([B,C,Nb,Xa],Kbs), [B,C,Nb,Xa]
   2.  B -> C :  Xb
   C computes Xc = h([C,S,Nc,Xb],Kcs), [C,S,Nc,Xb]
   3.  C -> S :  Xc
   4.  S -> C :  A, B, Kab ⊕ h(Na,Kas), {A,B,Na}Kab,
                 B, A, Kab ⊕ h(Nb,Kbs), {B,A,Nb}Kab,
                 B, C, Kbc ⊕ h(Nb,Kbs), {B,C,Nb}Kbc,
                 C, B, Kbc ⊕ h(Nc,Kcs), {C,B,Nc}Kbc
   5.  C -> B :  A, B, Kab ⊕ h(Na,Kas), {A,B,Na}Kab,
                 B, A, Kab ⊕ h(Nb,Kbs), {B,A,Nb}Kab,
                 B, C, Kbc ⊕ h(Nb,Kbs), {B,C,Nb}Kbc
   6.  B -> A :  A, B, Kab ⊕ h(Na,Kas), {A,B,Na}Kab
```

Attacks. This protocol is subject to an attack [96] that can be mounted by a dishonest participant. For example, assume that C is a malicious agent. He can intercept Kab ⊕ h(Nb,Kbs) and Kbc ⊕ h(Nb,Kbs) sent by S at step 4, and since C knows the session key Kbc, he can compute Kbc ⊕ Kab ⊕ h(Nb,Kbs) ⊕ Kbc ⊕ h(Nb,Kbs). Since this term is actually equal to Kab, the agent C learns a session key that should be shared only by A and B.

WIRED EQUIVALENT PRIVACY PROTOCOL

Summary: The Wired Equivalent Privacy (WEP) protocol, described in [1], is used to protect data during wireless transmission. To encrypt the message M, A applies the operator ⊕ to RC4(v,Kab) and [M,C(M)] where C(M) is the *integrity checksum* of the message M and RC4 is a function modeling the RC4 algorithm which is used to generate a *keystream* (*i.e.* a long sequence of pseudo-random bytes) from the initial vector v and the secret key Kab shared between A and B. To decrypt the received message, B computes RC4(v,Kab) and after applying exclusive or, he obtains [M,C(M)] and can verify that the checksum is correct.

Protocol specification.

```
A, B : principal
Kab :  symkey
RC4 :  message, symkey -> message
C :    message -> message
   1.  A -> B :   v, ([M,C(M)] ⊕ RC4(v,Kab))
```

Requirements We present below attacks given in [15] that require the following properties:

$$\texttt{C(x} \oplus \texttt{y)} = \texttt{C(x)} \oplus \texttt{C(Y)} \quad \text{(E5)}$$
$$\texttt{[x1,y1]} \oplus \texttt{[x2,y2]} = \texttt{[x1} \oplus \texttt{x2,y1} \oplus \texttt{y2]} \quad \text{(E6)}$$

According to [15], (E5) is a general property of CRC checksum.

Attacks. The first attack uses the fact that encrypting two messages `P1` and `P2` with the same initial vector `v` and with the same key `k` can reveal information. Indeed, we have the following equalities between the ciphers `C1` and `C2` and their associated plain text `P1` and `P2`:

$$\begin{aligned} \texttt{C1} \oplus \texttt{C2} &= \texttt{([P1,C(P1)]} \oplus \texttt{RC4(v,k))} \oplus \texttt{([P2,C(P2)]} \oplus \texttt{RC4(v,k))} \\ &= \texttt{([P1,C(P1)]} \oplus \texttt{[P2,C(P2)])} \oplus \texttt{(RC4(v,k)} \oplus \texttt{RC4(v,k))} && \text{(E1)(E2)} \\ &= \texttt{[P1,C(P1)]} \oplus \texttt{[P2,C(P2)]} && \text{(E3)(E4)} \end{aligned}$$

This allows an intruder who knows a plain text `P1` and its cipher `C1` to decrypt any cipher `C2`. Indeed, thanks to this equality, the intruder can easily get `[P2,C(P2)]` and obtain the plaintext `P2`.

The second attack allows the intruder controlled modifications to a cipher text without disrupting the checksum. Assume that the intruder has intercepted $\texttt{[M,C(M)]} \oplus \texttt{RC4(v,Kab)}$ and knows `D`. He can now obtain the cipher text associated to the message $\texttt{M} \oplus \texttt{D}$ by computing:

$$\begin{aligned} \texttt{([M,C(M)]} \oplus \texttt{RC4(v,Kab))} \oplus \texttt{[D,C(D)]} &= \texttt{RC4(v,Kab)} \oplus \texttt{([M,C(M)]} \oplus \texttt{[D,C(D)])} && \text{(E1)(E2)} \\ &= \texttt{RC4(v,Kab)} \oplus \texttt{[M} \oplus \texttt{D,C(M)} \oplus \texttt{C(D)]} && \text{(E6)} \\ &= \texttt{RC4(v,Kab)} \oplus \texttt{[M} \oplus \texttt{D,C(M} \oplus \texttt{D)]} && \text{(E5)} \end{aligned}$$

Notice that this attack can be applied without full knowledge of `M`. For example, to flip the first bit of `M`, the attacker can set `D = 100...0`. Now, if the intruder knows the plaintext `M` (and its associated cipher) he can generate the ciphertext associated to any message he wants.

GONG'S MUTUAL AUTHENTICATION PROTOCOL

Author(s): L. Gong (1989)
Summary: The protocol, presented in [51], aims at providing mutual authentication and distributing a fresh secret key `k`. It makes use of a trusted server `S` with which each of the two agents `A` and `B` shares a secret password (`Pa` and `Pb` respectively). As an alternative to encryption algorithms, this protocol uses the one-way functions `f` and `g`.

The principal `B` can extract, using the properties of exclusive or, the triple `[k, ha, hb]` from the message that he receives at step `3`, and check it by computing `g(k, ha, hb, Pb)`. Knowing `Pa` and after receiving `Ns`, `A` can build `f(Ns ,Na ,B, Pa)` to get the message `[k, ha, hb]`. Hence, she can verify the message `hb` sent by `B` at step `4` and send the message `ha` to `B` in order to prove her identity.

Protocol specification.

```
A, B, S :     principal
Na, Nb, Ns :  fresh number
Pa, Pb :      password
f :           message, message, message, message -> message
g :           message, message, message, message -> message
   1.  A -> B :  A, B, Na
   2.  B -> S :  A, B, Na, Nb
```

```
S computes [k, ha, hb] = f(Ns, Na, B, Pa)
3.  S -> B :  Ns, f(Ns, Nb, A, Pb) ⊕ [k, ha, hb], g(k, ha, hb, Pb)
4.  B -> A :  Ns, hb
5.  A -> B :  ha
```

3.4 Abelian Groups

The + symbol denotes the additive binary operation of Abelian groups (G,+). The properties of Abelian groups are:

$$
\begin{aligned}
x+(y+z) &= (x+y)+z && \text{(associativity)}\\
x+y &= y+x && \text{(commutativity)}\\
x+0 &= x && \text{(neutral element)}\\
x+(-x) &= 0 && \text{(inverse)}
\end{aligned}
$$

We present a protocol for computing a sum using these properties. Many protocols use the structure of Abelian groups in combination with other properties. Some of them are presented in Sections 3.7 and 3.8.

SALARY SUM

Summary: This protocol, described in [98], allows a group of people to compute the sum of their salaries without anyone declaring his own salary to the others. For the sake of simplicity, the protocol is given for four principals A, B, C and D.

Protocol specification.

```
A, B, C, D :     principal
PK, SK :         principal -> key (key pair)
Na :             fresh number
Sa, Sb, Sc, Sd : number
   1.  A ->    B     :   A, {Na + Sa}PK(B)
   2.  B ->    C     :   B, {Na + Sa + Sb}PK(C)
   3.  C ->    D     :   C, {Na + Sa + Sb + Sc}PK(D)
   4.  D ->    A     :   D, {Na + Sa + Sb + Sc + Sd}PK(A)
   5.  A -> B, C, D  :   Sa + Sb + Sc + Sd
```

Attacks. The protocol is flawed because there is no authentication. The following attack has been suggested by L. Bozga (private communication). A starts to establish a normal session with I. Then, the intruder I answers directly to A, impersonating D. Lastly, A answers, thinking that everybody has added his salary. Using the answer of A, the intruder I is able to learn A's salary.

```
1.    A   ->    I     :   A, {Na + Sa}PK(I)
4.  I(D) ->    A     :   D, {Na + Sa + Si + Si + Si}PK(A)
5.    A   -> I, C, D :   Sa + Si + Si + Si
```

There are many protocols using an addition or a subtraction between a nonce and a constant (generally the constant is equal to one) in a challenge response for example. This is the case of the Needham-Schroeder protocol with symmetric keys [87] presented in Section 3.6, the amended

Needham-Schroeder protocol [24], the Andrew Secure RPC protocol [97, 104] and the SPLICE/AS authentication protocol [54]. For all these protocols, it is possible to consider these operations either with all the power of Abelian groups or with only the function symbol `succ`. Note however that the `succ` primitives does not seem to be really necessary for some of the protocols like for instance the Andrew Secure RPC protocol.

3.5 Homomorphism

In this section, we consider operators that satisfy equalities of the form `f(g(x,y))=g(f(x),f(y))`. The homomorphism property is often used in many vote protocols. ElGamal [43], Paillier [89, 47], Goldwasser-Micali [49], Benaloh [8, 25], Naccache-Stern [81], Okamoto-Uchiyama [88] provide cryptographic primitives that verify the homomorphism property.

First, we show how the classical block chaining mechanism ECB, induces a homomorphism property, `{[x,y]}z=[{x}z,{y}z]`, that may be used to attack the Needham-Schroeder-Lowe protocol. Secondly, we present an attack on the TMN protocol relying on the equality `{x*y}z = {x}z*{y}z`. Finally, we describe the counting stage of an election protocol whose executability relies on the equality: `{x+y}z = {x}z * {y}z`.

NEEDHAM-SCHROEDER-LOWE PROTOCOL WITH ECB

Author(s): G. Lowe, R. Needham and M. Schroeder (1996)
Summary: Electronic Code Book (ECB) consists in partitioning the message into n-bit blocks and encrypting each of them separately, after having possibly padded the last block.

The famous Needham-Schroeder-Lowe [63] protocol, used for mutual authentication of two principals, is flawed if the encryption is implemented using ECB. At the end of the protocol, each principal is convinced to talk with the right principal, and to share the secrets `Na` and `Nb`. A basic property of ECB is that `{A,B,C}k={A}k,{B}k,{C}k` where the size of the blocks `A`, `B`, `C` is a multiple of the block length used by the cryptographic algorithm.

Protocol specification.

```
A, B :   principal
Na, Nb : fresh number
PK, SK : principal -> key (key pair)
   1.  A -> B :   {Na, A}PK(B)
   2.  B -> A :   {Na, Nb, B}PK(A)
   3.  A -> B :   {Nb}PK(B)
```

Attacks. An intruder can attack the protocol by playing two sessions of the protocol and extracting `{Na, Nb}PK(A)` from `{Na, Nb, B}PK(A)`. He reuses it together with `{I}PK(A)` to compute `{Na, Nb, I}PK(A)` and forces `A` to decrypt `Nb` for him.

```
   i.1.      A   ->  I   :   {Na, A}PK(I)
   ii.1.   I(A) ->  B   :   {Na, A}PK(B)
   ii.2.     B   -> I(A) :   {Na, Nb, B}PK(A)

   The intruder extracts {Na, Nb}PK(A) and computes {Na, Nb, I}PK(A).

   i.2.      I  -> A :   {Na, Nb, I}PK(A)
   i.3.      A  -> I :   {Nb}PK(I)
   ii.3.   I(A) -> B :   {Nb}PK(B)
```

Remark. This attack is possible only if ECB is used, and if the size of each nonce and principal name is a multiple of the maximal size of the cipher imposed by the encryption function. Note also that the ECB algorithm is not commonly used: it is known to provide only very weak security (semantic security can never be achieved in the ECB mode).

TMN

Author(s): M. Tatebayashi, N. Matsuzaki, and D.B. Newman (1989)
Summary: This is a symmetric key distribution protocol for digital mobile communication systems, such as cellular networks. The only trusted key is the public key of the server [106], see also [68]. For each session, the server verifies that the keys Ka and Kb have not been used in previous sessions.

Protocol specification.

```
A, B, S : principal
Ka, Kb :  fresh symkey
PK, SK :  principal -> key (keypair)
   1.  A -> S :   B, {Ka}PK(S)
   2.  S -> B :   A
   3.  B -> S :   A, {Kb}PK(S)
   4.  S -> A :   B, Kb ⊕ Ka
```

Requirements The property needed to perform the attack described below is a homomorphism relation between $*$ and the encryption algorithm. More precisely we need that {x}PK(S)∗{y}PK(S) = {x∗y}PK(S), where x and y range over messages.

Attacks. The attack presented in [102] is based on the fact that two intruders C and D can eavesdrop the message {Kb}PK(S) exchanged between A and B during a normal session i of the protocol. Sharing their keys Kc and Kd, the intruders C and D can fool the server during a new session ii. C sends {Kc}PK(S)∗{Kb}PK(S) (step 1) which is equal to {Kc∗Kb}PK(S) thanks to the homomorphism property. The server thinks that it is a fresh key and plays the rest of the protocol by decrypting {Kc∗Kb}PK(S) and sending Kd ⊕ (Kc∗Kb). Knowing Kd, the intruder C can deduce Kc*Kd. Then, since he knows Kc, he can recover Kb.

```
A, B, C, D, S : principal
Kb, Kc, Kd :    fresh symkey
PK, SK :        principal -> key (keypair)
i.3.    B -> S :   A, {Kb}PK(S)
ii.1.   C -> S :   D, {Kc}PK(S)∗{Kb}PK(S) (= {Kc∗Kb}PK(S))
ii.2.   S -> D :   C
ii.3.   D -> S :   C, {Kd}PK(S)
ii.4.   S -> C :   D, Kd ⊕ (Kc∗Kb)
```

Remark. Note that the simple attack consisting in sending {Kb}PK(S) at step 1 does not work since it is assumed that S can detect keys that have already been used in previous sessions.

Note also that the attack relies on the homomorphic property and not really on the properties of exclusive or. Indeed, a similar attack (see below) can be mounted when considering the abstract version of the protocol proposed in [80], where the last message `Kb ⊕ Ka` is replaced by `{Kb}Ka` to consider a more general method of encryption.

```
i.3.   B -> S :   A, {Kb}PK(S)
ii.1.  C -> S :   D, {Kc}PK(S)
ii.2.  S -> D :   C
ii.3.  D -> S :   C, {Kd}PK(S)*{Kb}PK(S)  (= {Kd*Kb}PK(S))
ii.4.  S -> C :   D, {Kd*Kb}Kc
```

MULTI-AUTHORITY SECRET BALLOT ELECTION PROTOCOL (COUNTING STAGE)

Author(s): R. Cramer, M. Franklin, B. Schoenmakers and M. Yung (1996)
Summary: This presentation is a simplification of one of the steps of the protocol presented by R. Cramer *et al.* in [33]. The goal of the protocol is to organize an election with the voters `Ai`. The voter `Ai` casts his vote `Vi = +1` or `Vi = -1` by posting a ballot to a *bulletin board*, *i.e* a broadcast channel with memory. The ballot is essentially the vote `Vi` encrypted by the public key of the authority `PK(S)` by using a probabilistic encryption algorithm. The encrypted message, written `crypt(Vi, ri, PK(S))`, does not reveal any information on the vote itself. It is ensured by an accompanying proof that the ballot indeed contains a valid vote. This proof of validity is described in more details in Section 3.8. When the deadline is reached, the authority `S` computes, using the homomorphism property, the encryption of the result of the election by computing the product of all encrypted messages received. The authority does not need to decrypt the vote of any single voter, which enables a quicker computation.

Protocol specification.

```
Ai, S :  principal
PK, SK : principal -> key (keypair)
crypt :  number, fresh number, key -> number
ri :     fresh number
Vi :     number
   i.  Ai -> S :   Bi = crypt(Vi, ri, PK(S))
      S computes B1 * ... * Bn
        decrypts with SK(S) and publishes the result
```

Requirements The main property required of the cryptographic algorithm is :

```
crypt(m1, r1, K) * crypt(m2, r2, K) = crypt(m1+m2, r, K)
```

Such a cryptographic algorithm can be obtained from the ElGamal cryptosystem [43] (for more details, see [33]).

3.6 Prefix Property

The prefix property is the ability of an intruder to get from an encrypted message the encryption of any of its prefixes: this property strongly depends on the encryption algorithm. For example, the ECB algorithm (presented in Section 3.5) and the CBC algorithm (presented in Section 2.3, Paragraph

Prefix Property) have this property. Indeed, assuming that the length of x is a multiple of the block length used in the algorithm, the intruder can deduce {x}z from {x,y}z.

We show how this property can be exploited to attack the Denning-Sacco symmetric key protocol and the Needham-Schroeder symmetric key protocol. In [7], this property is also used by S. Bellovin to mount an attack on IP security protocols.

Denning-Sacco Symmetric Key Protocol with CBC

Author(s): D. E. Denning and G. M. Sacco (1981)

Summary: This is a modified version [39] of the Needham-Schroeder symmetric key protocol with timestamps to fix the freshness flaw. It aims at distributing a shared symmetric key Kab using a trusted server and achieving mutual authentication, using symmetric key cryptography and timestamps.

Protocol specification.

```
A, B, S :  principal
Kas, Kbs : symkey
Kab :      fresh symkey
T :        timestamp

   1.  A -> S :  A, B
   2.  S -> A :  {B, Kab, T, {A, Kab, T}Kbs}Kas
   3.  A -> B :  {A, Kab, T}Kbs
```

Attacks. Y. Chevalier and L. Vigneron [23] present an attack on this protocol. It is based on the fact that messages 2 and 3 are of the same global form. It requires that the length of names of the agents, nonces and timestamps are a multiple of the block length used for encryption. The attack consists of the intruder impersonating B to start a session i with the server S. Using the answer {A, Kab, T, {B, Kab, T}Kas}Kbs of the server and the prefix property, the intruder is able to get the message {A, Kab, T}Kbs. Receiving this message, B accepts a new value for the symmetric key he shares with A, whereas A is not aware that a protocol run (session ii) took place.

```
   i.1.    I(B) ->   S   :  B, A
   i.2.      S  -> I(B) :  {A, Kab, T, {B, Kab, T}Kas}Kbs
   ii.3.   I(A) ->   B   :  {A, Kab, T}Kbs
```

Needham-Schroeder Symmetric Key Protocol with CBC

Author(s): R. Needham and M. Schroeder (1978)

Summary: This protocol aims at establishing a fresh shared symmetric key Kab and mutually authenticating the participants: in every session, the value of Kab has to be known only by the participants playing the roles of A, B and S in that session.

Messages 1 to 3 perform the distribution of the fresh shared symmetric key Kab and messages 4 and 5 are for mutual authentication of A and B. The operator succ is the increment operation.

Protocol specification.

```
A, B, S :     principal
Na, Nb :      fresh number
Kab :         fresh symkey
Kas, Kbs :    symkey
succ :        number -> number
   1.  A -> S :  A, B, Na
   2.  S -> A :  {Na, B, Kab, {Kab, A}Kbs}Kas
   3.  A -> B :  {Kab,A}Kbs
   4.  B -> A :  {Nb}Kab
   5.  A -> B :  {succ(Nb)}Kab
```

Attacks. Beyond other existing attacks, O. Pereira and J.-J. Quisquater [91] presented the following flaw, based on the prefix property. Suppose that the message {Na, B, Kab, {Kab, A}Kbs}Kas in step 2 has cipher text $C_0C_1C_2 \cdots C_n$ and that all components have length one block. Then {Na, B}Kas has cipher text $C_0C_1C_2$ and can be sent at step 3 (by the intruder eavesdropping B) to the agent A who plays the role of B in the session ii. Thus A can be fooled into accepting the publicly known nonce Na as a secret key shared with B.

```
   i.1.     A   ->   S   :  A, B, Na
   i.2.     S   ->   A   :  {Na, B, Kab, {Kab, A}Kbs}Kas
   ii.3.  I(B) ->   A   :  {Na, B}Kas
   ii.4.    A   -> I(B) :  {Na'}Na
   ii.5.  I(B) ->   A   :  {succ(Na')}Na
```

3.7 Abelian Groups and Modular Exponentiation

We consider Abelian group properties and a single property of modular exponentiation. The $\times$ symbol denotes multiplication, and exp denotes modular exponentiation.

$$\mathtt{x} \times (\mathtt{y} \times \mathtt{z}) = (\mathtt{x} \times \mathtt{y}) \times \mathtt{z} \quad \text{(E1)}$$
$$\mathtt{x} \times \mathtt{y} = \mathtt{y} \times \mathtt{x} \quad \text{(E2)}$$
$$\mathtt{x} \times 1 = \mathtt{x} \quad \text{(E3)}$$
$$\mathtt{x}^{-1} \times \mathtt{x} = 1 \quad \text{(E4)}$$
$$\mathtt{exp}(\mathtt{exp}(\mathtt{x},\mathtt{y}),\mathtt{z}) = \mathtt{exp}(\mathtt{x},\mathtt{y} \times \mathtt{z}) \quad \text{(E5)}$$

The equalities (E1), (E2), (E3) and (E5) listed above are required for the execution of the IKA.1 protocol.

IKA.1 Protocol

Author(s): M. Steiner, G. Tsudik and M. Waidner (1996)
Summary: The Initial Key Agreement protocol IKA.1, also known as GDH.2 [105, 6], aims at establishing a group key between a fixed number n of agents $A_1, \dots, A_n$. Each agent has a secret nonce N_i and knows the exponentiation base g. The group key is g raised to the product of all the nonces of the

group members. There are three kinds of agents: the initiator A_1 who starts the protocol, the last agent A_n who distributes to each agent the partial group key, and the others who constitute a chain from A_1 to A_n. A_1 begins with sending to his neighbor A_2 the message `exp(g,1)`,`exp(g,N1)`. Then, each agent $A_i (i = 2,\dots,n-1)$:

- receives a message $m = m_1,\dots,m_i$.
- stores the last component of m in $x = m_i$.
- raises to the N_i-th power all the components m_j, obtaining $m'_j = {m_j}^{N_i}$, $1 \leq j \leq i$ of the message m.
- creates with the message raised to the N_i-th power a new message $m' = m'_1,\dots,m'_{i-1},x,m'_i$.
- sends this new message m' to his neighbor A_{i+1}.

The last agent A_n receives a message composed of n parts, and raises to the N_n-th power all the components of the received message. The last component is the group key and the remaining part, which corresponds to partial group keys, is broadcast to all the agents by A_n. The other group members can compute the group key from this message by raising to the N_i-th power ith part of the message given by A_n. For the sake of simplicity, we describe the protocol for 3 participants.

Protocol specification.

```
A, B, C:     principal
Na, Nb, Nc:  fresh number
exp:         number, number -> number
   1.  A -> B    :  g,exp(g,Na)
   2.  B -> C    :  exp(g,Nb),exp(g,Na),exp(exp(g,Na),Nb)
   3.  C -> A, B :  exp(exp(g,Nb),Nc),exp(exp(g,Na),Nc)
```

Property. Details of the computation done by `A` to obtain the group key `exp(g,Na×Nb×Nc)`:

```
  exp(exp(exp(g,Nb),Nc),Na)
= exp(exp(g,Nb×Nc),Na)        (E5)
= exp(g,Na×Nb×Nc)             (E1)(E2)(E3)(E5)
```

Attacks. There is an attack [78, 52] that can be mounted by an active attacker. Indeed, the intruder can replay the first message to the initiator, impersonating the last agent. Hence `A` thinks that he has received a correct message from `C` and that the group key is `exp(exp(g,1),Na) = exp(g,Na)`, but this key is known by the intruder.

```
1.     A  -> B :  exp(g,1),exp(g,Na)
4.  I(C) -> A :  exp(g,1),exp(g,Na)
```

Remark. This protocol like the Diffie-Hellman protocol described before does not guarantee the authentication of the agents. Like it is proposed in [6] this protocol can be easily amended to avoid that an intruder take the place of an honest agent.

3.8 Abelian Groups and Extended Modular Exponentiation

The $+$, $\times$ and `exp` symbols denote respectively addition, multiplication and modular exponentiation. The corresponding equalities are listed below :

$$\mathtt{x} + (\mathtt{y} + \mathtt{z}) = (\mathtt{x} + \mathtt{y}) + \mathtt{z} \quad \text{(E1)} \qquad \mathtt{x} \times (\mathtt{y} \times \mathtt{z}) = (\mathtt{x} \times \mathtt{y}) \times \mathtt{z} \quad \text{(E9)}$$
$$\mathtt{x} + \mathtt{y} = \mathtt{y} + \mathtt{x} \quad \text{(E2)} \qquad \mathtt{x} \times \mathtt{y} = \mathtt{y} \times \mathtt{x} \quad \text{(E10)}$$
$$\mathtt{x} + \mathtt{0} = \mathtt{x} \quad \text{(E3)} \qquad \mathtt{x} \times \mathtt{1} = \mathtt{x} \quad \text{(E11)}$$
$$(-\mathtt{x}) + \mathtt{x} = \mathtt{0} \quad \text{(E4)} \qquad \mathtt{x} \times \mathtt{x}^{-1} = \mathtt{1} \quad \text{(E12)}$$

$$\mathtt{exp}(\mathtt{exp}(\mathtt{x},\mathtt{y}),\mathtt{z}) = \mathtt{exp}(\mathtt{x},\mathtt{y} \times \mathtt{z}) \quad \text{(E5)}$$
$$\mathtt{exp}(\mathtt{x},\mathtt{y}) \times \mathtt{exp}(\mathtt{x},\mathtt{z}) = \mathtt{exp}(\mathtt{x},\mathtt{y} + \mathtt{z}) \quad \text{(E6)}$$
$$\mathtt{exp}(\mathtt{x},\mathtt{0}) = \mathtt{1} \quad \text{(E7)}$$
$$\mathtt{exp}(\mathtt{x},\mathtt{1}) = \mathtt{x} \quad \text{(E8)}$$

All four protocols presented in this section use combination of addition, multiplication and modular exponentiation all together, but their execution may not need all of the equalities. This is the case of Schnorr's protocol, the MAKEP protocol and the SRP protocol. We specify explicitly for each of them which equalities are required. Only the last protocol (multi-authority secret ballot election protocol) requires all twelve equalities. Another protocol using all the properties of addition, multiplication and exponentiation is the AMP-3 protocol [61], a protocol designed to achieve authenticated key-exchange using human memorizable passwords. Let us also mention the Fortezza Key Exchange Algorithm [86] and the PAK-Z protocol [61, 71], which use both exclusive or and modular exponentiation.

SCHNORR'S PROTOCOL

Author(s): C. P. Schnorr (1991)
Summary: A zero-knowledge protocol is designed for convincing the verifier of the validity of a given statement, without releasing any knowledge beyond the validity of the statement. This concept was introduced in [50]. An overview can be found in [48]. We present the Schnorr protocol which is described by R. Cramer, I. Damgård and B. Schoenmakers in [31] and which uses this method. `A` wants to prove his identity to `B` by showing him that he knows `Sa` without revealing it.

Protocol specification.

```
A, B :          principal
Na, Nb :        fresh number
Sa :            private key
Pa = exp(g,Sa) :public key

   A chooses Na and computes a = exp(g,Na)
   1.  A -> B :   a
   B chooses Nb
   2.  B -> A :   Nb
   A computes r = Na + Nb × Sa
   3.  A -> B :   r
   B checks that exp(g, r) = a × exp(Pa,Nb)
```

Property. Details of the computation done by `B` at the last step of the protocol:

```
  a × exp(Pa,Nb)
= exp(g,Na) × exp(exp(g,Sa),Nb)
= exp(g,Na) × exp(g,Sa × Nb)  (E5)
= exp(g,Na + Sa × Nb)         (E6)
= exp(g, r)
```

Mutually Authentication Key Exchange Protocol (MAKEP)

Author(s): M. Jakobsson and D. Pointcheval (2001)
Summary: M. Jakobsson and D. Pointcheval propose a Mutually Authentication Key Exchange Protocol called MAKEP [56]. It has applications for devices with strict power consumption restrictions, such as wireless medical implants and contactless smart cards. G is a cyclic group of prime order q and g is a generator of G. The private key of A (resp. B) is denoted by Sa (resp. Sb). A (resp. B) has also a public key denoted by Pa (resp. Pb) such that Pa = exp(g, Sa) (resp. Pb = exp(g, Sb)). It is assumed that A knows B's public key, and vice versa. At the end of the protocol, A and B should be mutually authenticated and they should share the key $T = H_0$(exp(g,Sb),exp(g,Na′),exp(exp(g,Sb),Na′)) where Na′ is a fresh name generates by A during the session.

Protocol specification.

```
A, B :                           principal
Sa, Sb :                         private key
Pa = exp(g,Sa), Pb = exp(g,Sb) : public key
Na, Na′, Nb :                    fresh number
H0, H2 :                         number, number, number -> number
H1 :                             number -> number

A generates Na and Na′.
A computes H1(exp(g,Na)), T = H0(Pb,exp(g,Na′),exp(Pb,Na′))
   1.  A -> B :  exp(g,Na′), H1(exp(g,Na))
B chooses Nb and computes R′ = H2(Pb,exp(g,Na′),exp(exp(g,Na′),Sb))
   2.  B -> A :  R′,Nb
A verifies that R′ = H2(Pb,exp(g,Na′),exp(Pb,Na′))
A computes d = Na - Nb × Sa
   3.  A -> B :  d
B verifies H1(exp(g,Na)) = H1(exp(g,d) × exp(Pa,Nb))
B computes H0(Pb,exp(g,Na′),exp(exp(g,Na′),Sb)), the common session key.
```

Property. Details of the last check done by B:

```
  H1(exp(g,d) × exp(Pa,Nb))
= H1(exp(g,d) × exp(exp(g,Sa),Nb))
= H1(exp(g,d) × exp(g,Sa × Nb)     (E5)
= H1(exp(g,d + Sa ×Nb))            (E6)
= H1(exp(g,Na - Nb × Sa + Sa × Nb))
= H1(exp(g,Na + 0))                (E1)(E2)(E4)(E10)
= H1(exp(g,Na))                    (E3)
```

Attacks. There is an attack called Hijacking Attack found by D. Wong and A. Chan in [109] based on two parallel sessions played by the intruder to compromise the authentication of B.

Secure Remote Password Protocol

Author(s): T. Wu (1998)

Summary: The Secure Remote Password protocol, introduced by T. Wu in [110], is an authenticated key-exchange protocol designed to resist passive and active network adversaries even when used with relatively short, human-memorizable passwords. All values are computed modulo a large, safe integer N and g is a primitive root of $\mathbb{Z}/N\mathbb{Z}$. The agent A starts the protocol by sending his name to the server S, who generates a fresh number Ns1. Knowing the password Pa, both A and S compute x = H(Ns1,Pa). They exchange the values Va, Vs, and Ns3. Now they are both able to compute Ra = exp(Vs - exp(g,x),Na + Ns3 × x) = exp(Va × exp(exp(g,x),Ns3),Ns2) = Rs. They verify that they share the same number by sending two hashes: Ma = H(Va,Vs,H(Ra)) and H(Va,Ma,H(Rs)). At the end of a successful run, both sides share a secret session key K = H(Ra) = H(Rs).

Protocol specification.

```
A, S:               principal
Pa:                 password
Na, Ns1, Ns2, Ns3:  fresh number
H:                  number -> number

   1.  A -> S :  A
   2.  S -> A :  Ns1
   Both S and A compute x = H(Ns1,Pa) and A computes Va = exp(g,Na)
   3.  A -> S :  Va
   S computes Vs = exp(g,x) + exp(g,Ns2)
   4.  S -> A :  Vs, Ns3
   A computes Ra = exp(Vs - exp(g,x),Na + Ns3 × x)
   5.  A -> S :  Ma = H(Va,Vs,H(Ra))
   S computes Rs = exp(Va × exp(exp(g,x),Ns3),Ns2)
   S verifies Ma = H(Va,Vs,H(Rs))
   6.  S -> A :  Ms = H(Va,Ma,H(Rs))
   A verifies Ms = H(Va,Ma,H(Ra))
```

Property. Ra and Rs are equal modulo the equational theory:

```
   Ra
 = exp(Vs - exp(g,x),Na + Ns3 × x)
 = exp((exp(g,x) + exp(g,Ns2)) - exp(g,x),Na + Ns3 × x)
 = exp(exp(g,Ns2),Na + Ns3 × x)                         (E1)(E2)(E3)(E4)
 = exp(g, Ns2 × (Na + Ns3 × x))                         (E5)
 = exp(g,(Na + Ns3 × x) × Ns2)                          (E10)
 = exp(exp(g, Na + Ns3 × x), Ns2)                       (E5)
 = exp(exp(g, Na) × exp(g, Ns3 × x), Ns2)               (E6)
 = exp(exp(g, Na) × exp(g, x × Ns3), Ns2)               (E10)
 = exp(exp(g, Na) × exp(exp(g, x), Ns3), Ns2)           (E5)
 = exp(Va × exp(exp(g,x), Ns3), Ns2)
 = Rs
```

Remark. T. Wu [110] noticed that this protocol is subject to a two-for-one guessing attack: an attacker can verify two guesses per impersonating attempt. Even if it does not pose a significant practical security threat, he proposed a refinement of the protocol where the value `Vs` sent by the server is now equal to `3×exp(g,x) + exp(g,Ns2)`, to prevent such a two-for-one guessing flaw.

MULTI-AUTHORITY SECRET BALLOT ELECTION PROTOCOL (PROOF OF VALIDITY)

Author(s): R. Cramer, M. Franklin, B. Schoenmakers and M. Yung (1996)

Summary: This protocol, described in [32] and improved in [33], is a multi-authority secret ballot election scheme. We only present the proof of validity of a ballot `b` between a voter `A` and a verifier `B`. `G` is a cyclic group of prime order `q`, `g` and `h` are random group elements known by the participants. The voter `A` chooses his vote `v = 1` or `v = -1` and computes his ballot `b = exp(g,n)×exp(h,v)`. He computes the values `a1` and `a2`, as explained in Table 3, and sends them to `B`, who sends back a random number `c`. `A` finishes by computing (or choosing) `d1`, `d2`, `r1` and `r2` and sending them to `B`. Now the verifier `B` checks the validity of the ballot by verifying the relation between the two received messages.The details of all the computations of `A` are given in Table 3.

v = 1	v = -1
n, w, r1 ,d1 fresh number	n, w, r2, d2 fresh number
b ← exp(g,n)×exp(h,v)	b ← exp(g,n)×exp(h,v)
a1 ← exp(g,r1)× exp(b×h,-d1)	a1 ← exp(g,w)
a2 ← exp(g,w)	a2 ← exp(g,r2)× exp(b×h^{-1},-d2)
d2 ← c - d1	d1 ← c - d2
r2 ← w + n×d2	r1 ← w + n×d1

Table 3. Values computed by `A`

Protocol specification.

```
A, B :            principal
r1, r2, d1, d2 :  number
n, c, w :         fresh number

   1.  A -> B :  b, a1, a2
   2.  B -> A :  c
   3.  A -> B :  d1, d2, r1, r2

B verifies that: c = d1 + d2,
                 exp(g,r1) = a1×exp(b×h,d1),
                 exp(g,r2) = a2×exp(b×h^-1,d2)
```

Property. Details of the verifications of `B` are explained below for the case `v = 1`, the case `v = -1` is similar:

$$
\begin{array}{lll}
 & \texttt{d1 + d2} & \\
= & \texttt{d1 + (c - d1)} & \\
= & \texttt{c} & \text{(E1)(E2)(E3)(E4)}
\end{array}
$$

$$
\begin{array}{lll}
 & \texttt{a1}\times\texttt{exp(b}\times\texttt{h,d1)} & \\
= & \texttt{(exp(g,r1)}\times\texttt{exp(b}\times\texttt{h,-d1))}\times\texttt{exp(b}\times\texttt{h,d1)} & \\
= & \texttt{exp(g,r1)}\times\texttt{exp(b}\times\texttt{h,-d1+d1)} & \text{(E6)(E9)(E10)} \\
= & \texttt{exp(g,r1)}\times\texttt{exp(b}\times\texttt{h,0)} & \text{(E1)(E2)(E3)(E4)} \\
= & \texttt{exp(g,r1)}\times\texttt{1} & \text{(E7)} \\
= & \texttt{exp(g,r1)} & \text{(E11)}
\end{array}
$$

$$
\begin{array}{lll}
 & \texttt{a2}\times\texttt{exp(b}\times\texttt{h}^{-1}\texttt{,d2)} & \\
= & \texttt{exp(g,w)}\times\texttt{exp(b}\times\texttt{h}^{-1}\texttt{,d2)} & \\
= & \texttt{exp(g,w)}\times\texttt{exp((exp(g,n)}\times\texttt{exp(h,v))}\times\texttt{h}^{-1}\texttt{,d2)} & \\
= & \texttt{exp(g,w)}\times\texttt{exp((exp(g,n)}\times\texttt{exp(h,1))}\times\texttt{h}^{-1}\texttt{,d2)} & \\
= & \texttt{exp(g,w)}\times\texttt{exp((exp(g,n)}\times\texttt{ h)}\times\texttt{h}^{-1}\texttt{,d2)} & \text{(E8)} \\
= & \texttt{exp(g,w)}\times\texttt{exp(exp(g,n),d2)} & \text{(E9)(E10)(E11)(E12)} \\
= & \texttt{exp(g,w+n}\times\texttt{d2)} & \text{(E5)(E6)} \\
= & \texttt{exp(g,r2)} &
\end{array}
$$

Remark. The encryption used in the construction of the ballots has a homomorphic property in the sense as defined in Section 3.5, *i.e.* if `b1` and `b2` are encryptions of `v1` and `v2` then `b1`$\times$`b2` is an encryption of `v1+v2`.

3.9 Elliptic Curves

The famous Diffie-Hellman key exchange protocol presented in Section 3.2 and the protocols of Section 3.5 are based on group properties. We consider here elliptic curves which have the same properties but are more complex mathematical objects defined on a field. More details on elliptic curves can be found in [59, 79].

The symbols $+$ and $\times$ denote the two operations on a field K. $(K,+,\times)$ is a unitary commutative field if :

$$
\begin{array}{rl}
\mathtt{x}+(\mathtt{y}+\mathtt{z}) = (\mathtt{x}+\mathtt{y})+\mathtt{z} & \text{(E1)} \\
\mathtt{x}+\mathtt{y} = \mathtt{y}+\mathtt{x} & \text{(E2)} \\
(-\mathtt{x})+\mathtt{x} = \mathtt{0} & \text{(E3)} \\
\mathtt{x}+\mathtt{0} = \mathtt{x} & \text{(E4)}
\end{array}
\qquad
\begin{array}{rl}
(\mathtt{x}\times\mathtt{y})\times\mathtt{z} = \mathtt{x}\times(\mathtt{y}\times\mathtt{z}) & \text{(E5)} \\
\mathtt{x}\times\mathtt{y} = \mathtt{y}\times\mathtt{x} & \text{(E6)} \\
\mathtt{x}\times\mathtt{1} = \mathtt{x} & \text{(E7)} \\
\mathtt{x}\neq 0 \Rightarrow \mathtt{x}\times\mathtt{x}^{-1} = \mathtt{1} & \text{(E8)} \\
\mathtt{x}\times(\mathtt{y}+\mathtt{z}) = \mathtt{x}\times\mathtt{y}+\mathtt{x}\times\mathtt{z} & \text{(E9)}
\end{array}
$$

We may denote $a\times b$ as usually by ab for the product of elements of K. Two immediate consequences of (E5), (E6), (E7) and (E8) are the following properties:

$$
\begin{array}{rl}
\text{if } \mathtt{x}\times\mathtt{y}\neq 0 \Rightarrow (\mathtt{x}\times\mathtt{y})^{-1} = \mathtt{x}^{-1}\times\mathtt{y}^{-1} & \text{(E10)} \\
\text{if } \mathtt{x}\neq 0 \Rightarrow ((\mathtt{x})^{-1})^{-1} = \mathtt{x} & \text{(E11)}
\end{array}
$$

Given an elliptic curve (E), an operation $\dot{+}$ is defined on the points on the elliptic curve such that $(E) \cup O$ is an Abelian group, where O is an additional point. Thus $\dot{+}$ has the following properties:

$$\mathtt{P1} \dot{+} \mathtt{P2} = \mathtt{P2} \dot{+} \mathtt{P1} \quad \text{(EC1)}$$
$$\mathtt{P1} \dot{+} (\mathtt{P2} \dot{+} \mathtt{P3}) = (\mathtt{P1} \dot{+} \mathtt{P2}) \dot{+} \mathtt{P3} \quad \text{(EC2)}$$
$$\mathtt{P} \dot{+} (\dot{-}\mathtt{P}) = O \quad \text{(EC3)}$$
$$\mathtt{P} \dot{+} O = \mathtt{P} \quad \text{(EC4)}$$

We denote by `k.P` the sum of $\mathtt{P} \dot{+} \ldots \dot{+} \mathtt{P}$ (`k` times), for a natural number `k`. Given `P` and `k.P` on an elliptic curve and a fixed field, it is a hard problem to find `k`. This problem is called the discrete logarithm in the elliptic curve. We can deduce from (EC1)(EC2)(EC3)(EC4) the following equalities for constants `k` and `k'`:

- $\mathtt{k'.Q} \dot{+} \mathtt{k.Q} = \mathtt{(k'+k).Q}$ (EC5)
- $\mathtt{k'.(k.Q)} = \mathtt{(k'} \times \mathtt{k).Q}$ (EC6)

Many recent protocols use elliptic curves such as the PAK-EC protocol [70], a protocol achieving password authenticated key exchange. Here we only present two protocols: the Diffie-Hellman protocol and ECDSA [57].

The first example is the Diffie-Hellman protocol described in Section 3.2: Alice chooses a number `ka`, computes `ka.P` and sends it to Bob. Bob chooses a number `kb` and computes `kb.P` and sends it to Alice. Now, they share the common secret key $\mathtt{ka.(kb.P)} = \mathtt{(ka} \times \mathtt{kb).P} = \mathtt{kb.(ka.P)}$. Notice the attacks described in Section 3.2 can still be mounted.

Elliptic Curve Digital Signature Algorithm (ECDSA)

Author(s): D. Johnson and A. Menezes (1999)

Summary: We describe the first step of ECDSA [57] on a fixed elliptic curve and a fixed field known by all the principals. The agent `A` has a public key `sa.G`, where `G` is a point of the elliptic curve and `sa` is the private key of `A`. To sign a message `m`, the agent `A` creates a new key pair `(d,Q)` such that `Q = d.G` and $\mathtt{d} \in K$. `Q` is a point of the elliptic curve, made publicly available. `A` chooses a random number `k` and computes `k.G` and $\mathtt{s} = \mathtt{k}^{-1}\mathtt{(f(m)+dr)}$, where `r` is the first component of `k.G` and `f` is a function known by the two participants (SHA-1 in the original paper). If `k.G = 0` then `A` chooses another number `k` because otherwise anyone can take his identity. If `s = 0` then `A` also chooses an other number `k` because `0` does not have an inverse in the field K. Then `A` sends to `B` the message `m` and the signature `(r,s)`. The agent `B` checks the identity of `A` by computing the point $\mathtt{X} = \mathtt{(f(m)s}^{-1}\mathtt{).G} \dot{+} \mathtt{(rs}^{-1}\mathtt{).Q}$. If $\mathtt{X} = \mathtt{(X1,X2)} = O$ then `B` rejects the signature else otherwise accepts the signature provided `r = X1 mod n`.

Protocol specification.

```
A, B : principal
f :    number -> number
G,Q :  public point
k :    fresh number
sa,d : private key of A
m,n :  number
```

```
   A chooses a number k.
A computes k.G = (x1,y1), r = x1 mod n and s = k^-1(f(m)+dr).
   1.  A -> B :   m,(r,s)
```

Property. We give the details of the computation done by B to check the identity of A.

The agent B verifies:

$$
\begin{array}{ll}
\mathtt{X} & \\
= (\mathtt{f(m)s}^{-1}).\mathtt{G} \dotplus (\mathtt{rs}^{-1}).\mathtt{Q} & \\
= (\mathtt{f(m)(k}^{-1}\mathtt{(f(m)+dr))}^{-1}).\mathtt{G} \dotplus (\mathtt{r(k}^{-1}\mathtt{(f(m)+dr))}^{-1}).(\mathtt{d.G}) & \\
= (\mathtt{f(m)(f(m)+dr)}^{-1}\mathtt{k}).\mathtt{G} \dotplus (\mathtt{r(f(m)+dr)}^{-1}\mathtt{k}).(\mathtt{d.G}) & \text{(E6)(E10)(E11)} \\
= (\mathtt{f(m)(f(m)+dr)}^{-1}\mathtt{k} + \mathtt{r(f(m)+dr)}^{-1}\mathtt{kd}).\mathtt{G} & \text{(EC5)(EC6)} \\
= (\mathtt{f(m)(f(m)+dr)}^{-1} + \mathtt{dr(f(m)+dr)}^{-1}).(\mathtt{k.G}) & \text{(E9)(E5)(E6)(EC6)} \\
= (\mathtt{f(m)+\ dr)(f(m)+dr)}^{-1}).(\mathtt{k.G}) & \text{(E9)} \\
= 1.(\mathtt{k.G}) & \text{(E8) since } \mathtt{s} \neq \mathtt{0} \\
= \mathtt{k.G} & \text{(EC6)(E7)}
\end{array}
$$

We use the rules (E5-9) of the field plus all the rules over the elliptic curve.

3.10 DES Property

The DES encryption algorithm is not really used anymore [85]. However, it has the following property: the bitwise complement of an encrypted message is the encryption of the bitwise complement of the plaintext by the bitwise complement of the key. This can be described by the equation $\overline{\{\mathtt{x}\}_\mathtt{y}} = \{\overline{\mathtt{x}}\}_{\overline{\mathtt{y}}}$, where $\bar{\cdot}$ represents the bitwise complement operator.

Data Encryption System (DES)

Summary: This standard has been defined in [82] in 1977. The DES is a 64 bit block algorithm with a key of 56+8 bits. This symmetric encryption algorithm has been used in numerous protocols [41, 82–84, 101, 103]. We explain briefly the method of encryption (for more details see [37, 98]) and give an algebraic property on the DES.

Description of the encryption algorithm. Let m be a message of 64 bits. The algorithm to encrypt m by the key k of $56+8$ bits as follows:

1. First apply the initial permutation IP to m where IP is a fixed permutation. One obtains $x_0 = IP(\mathtt{m})$. Let L_0 and R_0 be the left and right halves of x_0: $x_0 = L_0R_0$.
2. Apply the following construction for $1 \leq i \leq 16$:
   $L_i = R_{i-1}$
   $R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$, where f is a tricky function using expansion, exclusive or, fixed substitutions called S-boxes and permutations and k_i are chains of 48 bits computed from k.
3. Finally apply the permutation IP^{-1} to get the encrypted text $\{m\}_k = IP^{-1}(R_{16}L_{16})$.

   To decrypt a message apply the same method in the reverse way.

Property. If $\{m\}_k$ is the message m encrypted with the key k by the DES algorithm then $\overline{\{m\}_k} = \{\overline{m}\}_{\overline{k}}$, where $\bar{x}$ is the bitwise complement of x, *e.g.* $\overline{0110} = 1001$. This property [37, 98] is well-known, but up to our knowledge, it has never been used to find an attack on any protocol.

3.11 Timestamps

Time plays a role in many cryptographic protocols. One of the simplest protocol with timestamps is the so-called Wide Mouthed Frog protocol.

WIDE MOUTHED FROG PROTOCOL

Author(s): M. Burrows (1989)
Summary: The protocol, described in [18], aims at guaranteeing the secrecy of a new shared key `Kab`: in each session, when receiving the last message, `B` must be convinced that the key `Kab` has been created by `A` in the same session on behalf of `S`.

Protocol specification.

```
A, S :     principal
Kas, Kbs : symkey
Kab :      fresh symkey
Ta, Ts :   timestamp
   1.  A -> S :  A, {Ta, B, Kab}Kas
   2.  S -> B :  {Ts, A, Kab}Kbs
```

Attacks. An attack on this protocol is described in [5]: `S` can update the timestamps from `A`'s time `Ta` to his time `Ts`. The effect is that the intruder can keep a key alive by using `S` as an oracle. This way, `I` can extend the lifetime of a (possibly compromised) key `Kab` as wanted. Note that this attack is not possible if we assume that messages contain enough redundancy so that the recipient can detect (and ignore) his own messages.

```
   i.1.      A   -> S :  A, {Ta, B, Kab}Kas
   i.2.      S   -> B :  {Ts, A, Kab}Kbs
   ii.1.   I(B)  -> S :  B, {Ts, A, Kab}Kbs
   ii.2.     S   -> A :  {Ts', B, Kab}Kas
   iii.1.  I(A)  -> S :  A, {Ts', B, Kab}Kas
   iii.2.    S   -> B :  {Ts", A, Kab}Kbs
   ....
```

4 Conclusion

In this survey, we have identified many algebraic properties that are particularly relevant for the analysis of cryptographic protocols. Since many recent protocols explicitly use lots of algebraic properties (addition, multiplication and modular exponentiation for example), cryptographic primitives cannot be considered as black boxes in such protocols. Even when the execution of a protocol does not require any special properties, we have seen that such protocols may be secure when assuming perfect cryptography while some flaws appear when looking closer at the implementation.

Many recent results consider some algebraic properties. However, the existing results presented in this survey have two main weaknesses. Firstly, they are mostly theoretical: very few practical implementations enable to automatically verify protocols with algebraic properties. Secondly, in most of the cases, each paper develops an *ad hoc* decision procedure for a particular property. Nothing is guaranteed if the property slightly changes. In the same manner, it is not clear what happens when combining several algebraic properties. That is why general approaches such as those of H. Comon-Lundh [27] or C. Meadows [74] are very interesting since they allow us to consider general classes of algebraic properties.

There also exist other ways to relax the perfect cryptography assumption, for example by modeling dictionary attacks [67]. We did not present such properties since we focused on algebraic properties but they are of course also relevant in the analysis of cryptographic protocols.

Acknowledgments

The authors would like to thank anonymous referees for their very helpful comments and Ralf Treinen to have read a preliminary version of this survey.

References

1. IEEE 802.11 Local and Metropolitan Area Networks: Wireless LAN Medium Acess Control (MAC) and Physical (PHY) Specifications, 1999.
2. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. In *Proc. 31st International Colloquium on Automata, Languages, and Programming (ICALP'04)*, vol. 3142 of *LNCS*, p. 46–58, Turku (Finland), 2004. Springer-Verlag.
3. R. Amadio and W. Charatonik. On name generation and set-based analysis in the Dolev-Yao model. In *Proc. International Conference on Concurrency Theory (CONCUR'02)*, vol. 2421 of *LNCS*, p. 499–514, Brno (Czech Republic), 2002. Springer-Verlag.
4. R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, 2002.
5. R. Anderson and R. Needham. Programming Satan's computer. In *Computer Science Today: Recent Trends and Developments*, vol. 1000 of *LNCS*, p. 426–440. Springer-Verlag, 1995.
6. G. Ateniese, M. Steiner, and G. Tsudik. New multiparty authentication services and key agreement protocols. *IEEE Journal of Selected Areas in Communications*, 18(4):628–639, 2000.
7. S. M. Bellovin. Problem areas for the IP security protocols. In *Proc. 6th USENIX Security Symposium*, p. 1–16, San José (California, USA), 1996. Usenix.
8. J. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret sharing. In *Proc. Advances in Cryptology (CRYPTO'86)*, vol. 263 of *LNCS*, p. 251–260, Santa Barbara (California, USA), 1987. Springer-Verlag.
9. S. Bistarelli, I. Cervesato, G. Lenzini, and F. Martinelli. Relating process algebras and multiset rewriting for security protocol analysis. In *Proc. 3rd Workshop on Issues in the Theory of Security (WITS'03)*, Warsaw (Poland), 2003.
10. B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*, p. 82–96, Cape Breton (Canada), 2001. IEEE Comp. Soc. Press.
11. B. Blanchet. *Cryptographic Protocol Verifier User Manual*, 2004.
12. B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In *Proc. 6th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'03)*, vol. 2620 of *LNCS*, p. 136–152, Warsaw (Poland), 2003. Springer-Verlag.
13. M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proc. 28th International Colloquium on Automata, Languages, and Programming (ICALP'01)*, vol. 2076 of *LNCS*, p. 667–681, Crete (Greece), 2001. Springer-Verlag.
14. M. Boreale and M. G. Buscemi. Symbolic analysis of crypto-protocols based on modular exponentiation. In *Proc. 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'03)*, vol. 2747 of *LNCS*, p. 269–278, Bratislava (Slovak Republic), 2003. Springer-Verlag.
15. N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: The insecurity of 802.11. In *Proc. 7th Annual International Conference on Mobile Computing and Networking (MOBICOM'01)*, p. 180–188, Rome (Italy), 2001. ACM Press.
16. L. Bozga, C. Ene, and Y. Lakhnech. A symbolic decision procedure for cryptographic protocols with time stamps. In *Proc. 15th International Conference on Concurrency Theory (CONCUR'04)*, LNCS, p. 177–192, London (England), 2004. Springer-Verlag.
17. J. Bull and D. J. Otway. The authentication protocol. Tech. Rep. DRA/CIS3/PROJ/CORBA/SC/1/CSM/436-04/03, Defence Research Agency, 1997.
18. M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Proc. 12th ACM Symposium on Operating System Principles (SOSP'89)*, p. 1–13, Litchfield Park (Arizona, USA), 1989. ACM Press.

19. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *Proc. of 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*, p. 261–270, Ottawa (Canada), 2003. IEEE Comp. Soc. Press.
20. 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'04)*, p. 53–63, Cork (Ireland), 2004.
21. Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, and L. Vigneron. Deciding the security of protocols with Diffie-Hellman exponentiation and product in exponents. In *Proc. 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'03)*, vol. 2914 of *LNCS*, p. 124–135, Mumbai (India), 2003. Springer-Verlag.
22. Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, and L. Vigneron. Extending the Dolev-Yao intruder for analyzing an unbounded number of sessions. In *Proc. 17th International Workshop in Computer Science Logic (CSL'03)*, vol. 2803 of *LNCS*, p. 128–141, Vienna (Austria), 2003. Springer-Verlag.
23. Y. Chevalier and L. Vigneron. Automated unbounded verification of security protocols. In *Proc. 14th International Conference on Computer Aided Verification (CAV'02)*, vol. 2404 of *LNCS*, p. 324–337, Copenhagen (Denmark), 2002. Springer-Verlag.
24. J. Clark and J. Jacob. A survey of authentication protocol literature. `http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps`, 1997.
25. J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proc. 26th Annual Symposium on Foundations of Computer Science (FOCS'85)*, p. 372–382, Portland (Oregon, USA), 1985. IEEE Comp. Soc. Press.
26. H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. Research Report LSV-01-13, Laboratoire Spécification and Vérification, ENS de Cachan, France, 2001.
27. H. Comon-Lundh. Intruder theories (ongoing work). In *Proc. 7th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'04)*, vol. 2987 of *LNCS*, p. 1–4, Barcelona (Spain), 2004. Springer-Verlag.
28. H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proc. 14th International Conference on Rewriting Techniques and Applications (RTA'2003)*, vol. 2706 of *LNCS*, p. 148–164, Valencia (Spain), 2003. Springer-Verlag.
29. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. of 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*, p. 271–280, Ottawa (Canada), 2003. IEEE Comp. Soc. Press.
30. H. Comon-Lundh and R. Treinen. Easy intruder deductions. In *Verification: Theory & Practice, Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday*, vol. 2772 of *LNCS*, p. 225–242. Springer-Verlag, 2003.
31. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proc. 14th Annual International Cryptology Conference (CRYPTO'94)*, vol. 963 of *LNCS*, p. 174–187, Santa Barbara (California, USA), 1994. Springer-Verlag.
32. R. Cramer, M. K. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In *Proc. International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'96)*, vol. 1070 of *LNCS*, p. 72–83, Zaragoza (Spain), 1996. Springer-Verlag.
33. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proc. International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'97)*, vol. 1233 of *LNCS*, p. 103–118, Konstanz (Germany), 1997. Springer-Verlag.
34. R. M. K. D. Dolev, S. Even. On the security of ping-pong protocols. In *Proc. Advances in Cryptology (CRYPTO'82)*, p. 177–186, Santa Barbara (California, USA), 1983.
35. M. D. Davis and E. J. Weyuker. *Computability, complexity and languages*, chap. 7, p. 128–132. Computer Science and Applied Mathematics. Academic Press, 1983.
36. S. Delaune and F. Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In *Proc. 11th ACM Conference on Computer and Communications Security (CCS'04)*, p. 278–287, Washington, USA, 2004. ACM.
37. H. Delfs and H. Knebl. *Introduction to Cryptography : Principles and Applications*. Springer-Verlag, 2002.
38. G. Delzanno and P. Ganty. Automatic verification of time sensitive cryptographic protocols. In *Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, vol. 2988 of *LNCS*, p. 342–356, Barcelona (Spain), 2004. Springer-Verlag.
39. D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, 1981.
40. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Society*, 22(6):644–654, 1976.

41. W. Diffie and M. Hellman. Exhaustive cryptanalysis of the NBS Data Encryption Standard. *IEEE Computer*, 10:74–84, 1977.
42. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. Workshop on Formal Methods and Security Protocols (FMSP'99)*, Trento (Italy), 1999.
43. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proc. Advances in Cryptology (CRYPTO'84)*, vol. 196 of *LNCS*, p. 10–18, Santa Barbara (California, USA), 1985. Springer-Verlag.
44. N. Evans and S. Schneider. Analysing time dependent security properties in CSP using PVS. In *Proc. 6th European Symposium on Research in Computer Security (ESORICS'00)*, vol. 1895 of *LNCS*, p. 222–237, Toulouse (France), 2000. Springer-Verlag.
45. S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. In *Proc. 24th Annual Symposium on Foundations of Computer Science (FOCS'83)*, p. 34–39, Tucson (Arizona, USA), 1983. IEEE Comp. Soc. Press.
46. S. Even, O. Goldreich, and A. Shamir. On the security of ping-pong protocols when implemented using the RSA. In *Proc. Advances in Cryptology (CRYPTO'85)*, vol. 218 of *LNCS*, p. 58–72, Santa Barbara (California, USA), 1986. Springer-Verlag.
47. P.-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *Proc. 4th International Conference on Financial Cryptography (FC'00)*, vol. 1962 of *LNCS*, p. 90–104, Anguilla (British West Indies), 2001. Springer-Verlag.
48. O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.
49. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
50. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proc. 17th annual ACM Symposium on Theory of Computing*, p. 291–304. ACM Press, 1985.
51. L. Gong. Using one-way functions for authentication. *SIGCOMM Computer Communication*, 19(5):8–11, 1989.
52. J. Goubault-Larrecq, M. Roger, and K. N. Verma. Abstraction and resolution modulo AC: How to verify Diffie-Hellman-like protocols automatically. *Journal of Logic and Algebraic Programming*, 2004. To appear.
53. H. Hüttel. Deciding framed bisimulation. In *Proc. 4th International Workshop on Verification of Infinite-State Systems (INFINITY'02)*, p. 1–20, Brno (Czech Republic), 2002.
54. T. Hwang and Y.-H. Chen. On the security of SPLICE/AS : The authentication system in WIDE Internet. *Information Processing Letters*, 53(2):91–101, 1995.
55. F. Jacquemard, M. Rusinowitch, and L. Vigneron. Compiling and verifying security protocols. In *Proc. of 7th International Conference on Logic for Programming and Automated Reasoning (LPAR'00)*, vol. 1955 of *LNCS*, p. 131–160, Reunion Island (France), 2000. Springer-Verlag.
56. M. Jakobsson and D. Pointcheval. Mutual authentication for low-power mobile devices. In *Proc. 5th International Conference on Financial Cryptography (FC'01)*, vol. 2339 of *LNCS*, p. 178–195, Grand Cayman (British West Indies), 2002. Springer-Verlag.
57. D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ECDSA). Tech. Rep. CORR 99-34, Univ. of Waterloo, Canada, 1999.
58. D. Kapur, P. Narendran, and L. Wang. An E-unification algorithm for analyzing protocols that use modular exponentiation. In *Proc. 14th International Conference on Rewriting Techniques and Applications (RTA'2003)*, vol. 2706 of *LNCS*, p. 165–179, Valencia (Spain), 2003. Springer-Verlag.
59. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.
60. S. Kremer and M. Ryan. Analysing the vulnerability of protocols to produce known-pair and chosen-text attacks. In *Proc. 2nd International Workshop on Security Issues in Coordination Models, Languages and Systems (SecCo'04)*, ENTCS, London, UK, 2005. Elsevier Science Publishers. To appear.
61. T. Kwon. Summary of AMP (authentication and key agreement via memorable passwords). Draft Document, August 2003.
62. P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for AC-like equational theories with homomorphisms. In *Proc. 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, LNCS, Nara, Japan, 2005. Springer. To appear.
63. G. Lowe. An attack on the Needham-Schroeder public key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.
64. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proc. 2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, vol. 1055 of *LNCS*, p. 147–166, Berlin (Germany), 1996. Springer-Verlag.
65. G. Lowe. Casper: A compiler for the analysis of security protocols. In *Proc. 10th Computer Security Foundations Workshop (CSFW'97)*, p. 18–30, Rockport (Massachusetts, USA), 1997. IEEE Comp. Soc. Press.
66. G. Lowe. Towards a completeness result for model checking of security protocols. In *Proc. 11th Computer Security Foundations Workshop (CSFW'98)*, p. 96–106, Rockport (Massachusetts, USA), 1998. IEEE Comp. Soc. Press.

67. G. Lowe. Analysing protocols subject to guessing attacks. In *Proc. of the Workshop on Issues in the Theory of Security (WITS'02)*, Portland (Oregon, USA), 2002.
68. G. Lowe and A. W. Roscoe. Using CSP to detect errors in the TMN protocol. *IEEE Transactions on Software Engineering*, 23(10):659–669, 1997.
69. C. Lynch and C. Meadows. On the relative soundness of the free algebra model for public key encryption. In *Proc. 4th Workshop on Issues in the Theory of Security (WITS'04)*, p. 81–95, Barcelona (Spain), 2004.
70. P. MacKenzie. More efficient password-authenticated key exchange. In *Proc. Cryptographer's Track at RSA Conference*, vol. 2020 of *LNCS*, p. 361–377, San Francisco (California, USA), 2001. Springer-Verlag.
71. P. McKenzie, S. Patel, and R. Swaminathan. Password-authenticated key exchange based on RSA. In *Proc. 6th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'00)*, vol. 1976 of *LNCS*, p. 599–613, Kyoto (Japan), 2000. Springer-Verlag.
72. C. Meadows. Language generation and verification in the NRL protocol analyzer. In *Proc. 9th Computer Seurity Foundation Workshop (CSFW'96)*, p. 48–62, Kenmare (Ireland), 1996. IEEE Comp. Soc. Press.
73. C. Meadows. Extending formal cryptographic protocol analysis techniques for group protocols and low-level cryptographic primitives. In *Proc. 1st Workshop on Issues in the Theory of Security (WITS'00)*, p. 87–92, Geneva (Switzerland), 2000.
74. C. Meadows. Towards a hierarchy of cryptographic protocol models. In *Proc. Workshop on Formal Methods in Security Engineering (FMSE'03)*, Washington (USA), 2003.
75. J. Millen. On the freedom of decryption. *Information Processing Letters*, 86(6):329–333, 2003.
76. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*. ACM Press, 2001.
77. J. Millen and V. Shmatikov. Symbolic protocol analysis with products and Diffie-Hellman exponentiation. In *Proc. 16th Computer Security Foundation Workshop (CSFW'03)*, p. 47–62, Pacific Grove (California, USA), 2003. IEEE Comp. Soc. Press.
78. J. K. Millen and G. Denker. CAPSL and MuCAPSL. *Journal of Telecommunications and Information Technology*, 4:16–27, 2002.
79. V. S. Miller. Use of elliptic curves in cryptography. In *Proc. Advances in Cryptology (CRYPTO'85)*, vol. 218 of *LNCS*, p. 417–426, Santa Barbara (California, USA), 1986. Springer-Verlag.
80. J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murφ. In *Proc. 1997 IEEE Symposium on Security and Privacy*, p. 141–151, Oakland (California, USA), 1997. IEEE Comp. Soc. Press.
81. D. Naccache and J. Stern. A new public-key cryptosystem. *Proc. International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'97)*, 1233:27–37, 1997.
82. National Bureau of Standards. *FIPS Publication 46: Announcing the Data Encryption Standard*, January 1977.
83. National Bureau of Standards. *FIPS Publication 81: DES Modes of Operation*, December 1980.
84. National Bureau of Standards. *FIPS Publication 46-1: Data Encryption Standard*, January 1988.
85. National Institute of Standards and Technology. Announcing proposed withdrawal of Federal Information Processing Standard (FIPS) for the Data Encryption Standard (DES) and request for comments. *Federal Register*, 69(142):44509–44510, 2004.
86. National Security Agency. *SKIPJACK and KEA algorithm secification,Version 2.0*, 1998.
87. R. Needham and M. Schroeder. Using encryption for authentification in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.
88. T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Proc. International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'98)*, vol. 1403, p. 308–318, Helsinki (Finland), 1998. Springer-Verlag. LNCS.
89. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'99)*, vol. 1592 of *LNCS*, p. 223–238, Prague (Czech Republic), 1999. Springer-Verlag.
90. L. Paulson. Mechanized proofs for a recursive authentication protocol. In *Proc. 10th Computer Security Foundations Workshop (CSFW'97)*, p. 84–95, Rockport (Massachusetts, USA), 1997. IEEE Comp. Soc. Press.
91. O. Pereira and J.-J. Quisquater. On the perfect encryption assumption. In *Proc. 1st Workshop on Issues in the Theory of Security (WITS'00)*, p. 42–45, Geneva (Switzerland), 2000.
92. O. Pereira and J.-J. Quisquater. A security analysis of the cliques protocols suites. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*, p. 73–81, Cape Breton (Canada), 2001.
93. R. Ramanujam and S. P. Suresh. A decidable subclass of unbounded security protocols. In *Proc. IFIP Workshop on Issues in the Theory of Security (WITS'03)*, p. 11–20, Warsaw (Poland), 2003.
94. R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable for unbounded nonces as well. In *Proc. 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'03)*, vol. 2914 of *LNCS*, p. 363–374, Mumbai (India), 2003. Springer-Verlag.

95. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*, p. 174–190, Cape Breton (Canada), 2001. IEEE Comp. Soc. Press.
96. P. Y. A. Ryan and S. A. Schneider. An attack on a recursive authentication protocol: A cautionary tale. *Information Processing Letters*, 65(1):7–10, 1998.
97. M. Satyanarayanan. Integrating security in a large distributed system. *ACM Transactions on Computer Systems*, 7(3):247–280, 1989.
98. B. Schneier. *Applied Cryptography*. Wiley, second edition, 1996.
99. H. Seidl and K. N. Verma. Flat and one-variable clauses: Complexity of verifying cryptographic protocols with single blind copying. In *Proc. of 11thth International Conference on Logic for Programming and Automated Reasoning (LPAR'04)*, LNCS, p. To appear, Montevideo (Uruguay), 2005. Springer-Verlag.
100. V. Shmatikov. Decidable analysis of cryptographic protocols with products and modular exponentiation. In *Proc. 13th European Symposium On Programming (ESOP'04)*, vol. 2986 of *LNCS*, p. 355–369, Barcelona (Spain), 2004. Springer-Verlag.
101. V. Shoup and A. Rubin. Session key distribution using smart cards. In *Proc. International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'96)*, vol. 1070 of *LNCS*, p. 321–330, Zaragoza (Spain), 1996. Springer-Verlag.
102. G. Simmons. Cryptoanalysis and protocol failures. *Communications of the ACM*, 37(11):56–65, 1994.
103. M. E. Smid and D. K. Branstad. The Data Encryption Standard: Past and future. *Proc. IEEE*, 76(5):550–559, 1988.
104. Security protocols open repository. `http://www.lsv.ens-cachan.fr/spore/index.html`.
105. M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman key distribution extended to groups. In *Proc. 3rd ACM Conference on Computer and Communications in Security, (CCS'96)*, p. 31–37. ACM Press, 1996.
106. M. Tatebayashi, N. Matsuzaki, and D. B. Newman. Key distribution protocol for digital mobile communication systems. In *Proc. 9th Annual International Cryptology Conference (CRYPTO'89)*, vol. 435 of *LNCS*, p. 324–333, Santa Barbara (California, USA), 1989. Springer-Verlag.
107. M. Turuani. *Sécurité des protocoles cryptographiques: décidabilité et complexité*. PhD thesis, Université Henri Poincaré, Nancy (France), 2003.
108. K. N. Verma. Two-way equational tree automata for AC-like theories: Decidability and closure properties. In *Proc. 14th International Conference on Rewriting Techniques and Applications (RTA'03)*, vol. 2706 of *LNCS*, p. 180–196, Valencia (Spain), 2003. Springer-Verlag.
109. D. S. Wong and A. H. Chan. Efficient and mutually authenticated key exchange protocol for low power computing devices. In *Proc. 7th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'01)*, vol. 2248 of *LNCS*, p. 272–289, Gold Coast (Australia), 2001. Springer-Verlag.
110. T. Wu. The secure remote password protocol. In *Proc. 1998 Internet Society Network and Distributed System Security Symposium*, p. 97–111, San Diego (California, USA), 1998.