A survey of symbolic methods for establishing equivalence-based properties in cryptographic protocols

Stéphanie Delaune[a], Lucca Hirschi[b]

[a] *CNRS/IRISA, Rennes, France*
[b] *LSV, CNRS & ENS Cachan, France*

Abstract

Cryptographic protocols aim at securing communications over insecure networks such as the Internet, where dishonest users may listen to communications and interfere with them. A secure communication has a different meaning depending on the underlying application. It ranges from the confidentiality of a data to *e.g.* verifiability in electronic voting systems. Another example of a security notion is *privacy*.

Formal symbolic models have proved their usefulness for analysing the security of protocols. Until quite recently, most results focused on trace properties like confidentiality or authentication. There are however several security properties, which cannot be defined (or cannot be naturally defined) as trace properties and require a notion of behavioural equivalence. Typical examples are anonymity, and privacy related properties. During the last decade, several results and verification tools have been developed to analyse equivalence-based security properties.

We propose here a synthesis of decidability and undecidability results for equivalence-based security properties. Moreover, we give an overview of existing verification tools that may be used to verify equivalence-based security properties.

Keywords: cryptographic protocols, symbolic models, privacy-related properties, behavioural equivalence

1. Introduction

Security protocols are widely used to secure transmissions in various types of networks (*e.g.,* web, wireless devices, *etc.*). They are (often small) concurrent programs relying on cryptographic primitives. The security properties

they should achieve are multiple and depend on the context in which they are used. The main problem they have to cope with is to protect communication that are done through insecure, public, channels like the Internet, where dishonest users may listen to communications and interfere with them. This explains why they are notoriously difficult to design and hard to analyse by hand. Actually, many protocols have been shown to be flawed several years after their publication (and deployment). Given the very sensitive contexts in which they are used, establishing the security of these protocols is a very relevant research goal with important economic and societal consequences.

Two main distinct approaches have emerged, starting with the early 1980's attempt of [1], to ground security analysis of protocols on firm, rigorous mathematical foundations. These two approaches are known as the *computational approach* and the *symbolic approach*.

The computational approach models messages as bit-strings; agents and the attacker as probabilistic polynomial time machines; whereas security properties are defined using games played by the attacker who has to be able to distinguish the protocol from an idealised version of it (with a non negligible probability). It is generally acknowledged that security proofs in this model offer powerful security guarantees. A serious downside of this approach however is that even for small protocols, proofs are usually long, difficult, tedious, and highly error prone. Moreover, due to the high complexity of such a model, automating such proofs is a very complex problem that is still in its infancy (see *e.g.* [2]).

By contrast, the symbolic approach, which is the one targeted by this survey, makes strong assumptions on cryptographic primitives (*i.e.,* black-boxed cryptography assumption) but fully models agents' interactions and algebraic properties of these primitives. For instance, symmetric encryption and decryption are modelled as function symbols enc and dec along with the equations $\mathsf{dec}(\mathsf{enc}(m, k), k) = m$. This means that, without the corresponding key k, it is simply impossible to get back the plaintext m from the cipher-text $\mathsf{enc}(m, k)$. This does not mean however that protocols relying on these primitives are necessarily secure. There can still remain some *logical attacks* like *e.g.* a man-in-the-middle attack or a reflection attack. Although less precise, this symbolic approach benefits from automation and can thus target more complex protocols than those analysed using the computational approach. Moreover, a line of work known as *computational soundness* aims at spanning the gap between these two approaches by establishing that, in

some cases, security guarantees in the symbolic model imply security guarantees in the computational one. This line has been initiated by Abadi and Rogaway [3] and has received much attention since then (see [4] for a survey on computational soundness).

Until the early 2000s, most works from the symbolic approach were focusing on *trace properties*, that is, statements that something bad never occurs on any execution trace of a protocol. Secrecy and authentication are typical examples of trace properties: a data remains confidential if, for any execution, the attacker is not able to produce the data from its observations. But many other properties like strong secrecy, unlinkability or anonymity are not defined as trace properties. These properties are usually defined as the fact that an observer cannot distinguish between two situations, and require a notion of *behavioural equivalence*. Roughly, two protocols are equivalent if an attacker cannot observe whether he is interacting with one or the other. In this survey, we shall focus on equivalence-based security properties.

There exist other approaches out of the scope of this survey that do not strictly follow the symbolic approach nor the computational one but are able to verify notions of behavioural equivalence. A recent approach proposes to define a computationally complete symbolic attacker by axiomatizing what the attacker can *not* do [5]. This approach has been recently extended to deal with a notion of behavioural equivalence [6]. Another work proposed semi-automatic proof of vote privacy using type-based verification [7]. This has been done using the tool Rf^*, where protocols are modelled using code-based cryptographic abstractions and security properties are encoded as *refinement types* [8]. Security is achieved by type checking the protocol.

Outline. In Section 2, we give an informal presentation of different cryptographic primitives after which we describe the Basic Access Control (BAC) protocol from the e-passport application, and some of its logical attacks. Section 3 describes various security properties that one may want to verify. In Section 4, we give a formal model, following the symbolic approach, for messages, protocols and equivalence properties. Section 5 is dedicated to the existing methods and tools for verifying equivalence-based properties. We conclude in Section 6.

2. What is a cryptographic protocol?

A cryptographic protocol can be seen as a list of rules that describe executions; these rules specify the emissions and receptions of messages by the actors of the protocols called *agents*. These protocols use as basic building blocks cryptographic primitives such as symmetric/asymmetric encryptions, signatures, and hash functions. For a long time, it was believed that designing a strong encryption scheme was sufficient to ensure secure message exchanges. Starting from the 1980's, researchers understood that even with perfect encryption schemes, message exchanges were still not necessarily secure. This fact will be illustrated in Section 2.2, but we first briefly explain the most standard cryptographic primitives together with their fundamental properties.

2.1. Cryptographic primitives

Cryptographic primitives provide fundamental properties and are used to develop more complex tools called cryptographic protocols, which guarantee one or more high-level security properties.

Symmetric encryption. Symmetric cryptography refers to encryption methods in which both the sender and the receiver share the same key. For instance, the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are symmetric encryption schemes which have been designated cryptography standards by the US government in 1976 and 2002 respectively.

A significant disadvantage of symmetric ciphers is the key management necessary to use them securely. Each distinct pair of communicating parties must share a different key. Therefore, the number of required keys increases as the square of the number of network members, which very quickly requires complex key management schemes to keep them all straight and secret. The difficulty of securely establishing a secret key between two communicating parties, when a secure channel does not already exist between them, also presents a chicken-and-egg problem which is a considerable practical obstacle for the use of cryptography in the real world. This is why the recourse to asymmetric cryptography is so popular for key establishment protocols that aim to establish a fresh symmetric key between two parties.

Asymmetric encryption. In 1976, Diffie and Hellman proposed the notion of public key cryptography, in which two different but mathematically related keys are used – a public key and a private key [9]. A public key system is constructed, in such a way that calculation of one key (the 'private key') is computationally infeasible from the other (the 'public key'), even though they are necessarily related. In public key cryptosystems, the public key may be freely distributed, while its associated private key must remain secret. The public key is typically used for encryption, while the private key is used for decryption. Diffie and Hellman showed that public key cryptography was possible[1] by presenting the Diffie-Hellman key exchange protocol [9]. In 1978, Rivest, Shamir, and Adleman invented RSA, another public key cryptosystem [10] which has established itself as the main standard.

Digital signature. Over the same period, signature schemes have also been proposed. A digital signature is a mathematical scheme for demonstrating the authenticity of a digital message or of a document. It gives the recipient a reason to believe that the message was created by a known sender, that the sender cannot deny having sent the message (authentication and non-repudiation), and that the message was not altered while in transit (integrity). Digital signatures are commonly used for software distribution, key management, financial transactions, *etc.*

Hash function. A hash function takes a message of any length as input, and outputs a short, fixed length hash. Hash functions have many information security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. They can also be used as checksums to detect accidental data corruption. For good hash functions, an attacker cannot find two messages that produce the same hash. Message authentication codes are much like cryptographic hash functions, except that a secret key can be used to authenticate the hash value upon receipt.

This list of cryptographic primitives is not exhaustive, and modern protocols often rely on less standard cryptographic primitives, such as blind signature, homomorphic encryption, trapdoor bit commitment.

[1] Diffie and Hellman have been rewarded by the ACM Turing Award in 2015 for having laid the foundations of asymmetric encryption.

2.2. An example: the BAC protocol

For the purpose of illustration, we consider the Basic Access Control (BAC) protocol used in the e-passport application. An e-passport is a paper passport with an RFID chip that stores the critical information printed on the passport. The International Civil Aviation Organisation (ICAO) standard specifies the communication protocols that are used to access this information [11]. We do not plan to describe all the protocols that are specified in the standard. Instead, we shall concentrate only on the BAC protocol following the modelling proposed in [12].

The information stored in the chip is organised in data groups (dg_1 to dg_{19}). For example, dg_5 contains a JPEG copy of the displayed picture, and dg_7 contains the displayed signature. The verification key $\mathsf{vk}(sk_P)$ of the passport, together with its certificate $\mathsf{sign}(\mathsf{vk}(sk_P), sk_{DS})$ issued by the Document Signer Authority, is stored in dg_{15}. The corresponding signing key sk_P is stored in a tamper resistant memory, and cannot be read or copied. For authentication purposes, a hash of all the data groups, together with a signature on this hash value issued by the Document Signer Authority, are stored in a separate file, the Security Object Document:

$$sod \stackrel{\mathsf{def}}{=} \langle \mathsf{sign}(\mathsf{h}(dg_1, \ldots, dg_{19}), sk_{DS}),\ \mathsf{h}(dg_1, \ldots, dg_{19}) \rangle.$$

The ICAO standard specifies several protocols through which this information can be accessed. In particular, read access to the data on the passport is protected by the BAC protocol.

The BAC protocol is a password-based authenticated key exchange protocol (PAKE) whose security relies on two master keys, namely ke and km, which are derived from a password of low entropy optically retrieved from the passport by the reader before executing the protocol. Through the BAC protocol, the reader and the passport agree on a key seed $xkseed$ that is then used to generate an encryption session key as well as a MAC session key for the next protocols. Following the description given in Figure 1, the protocol works as follows:

1. The reader sends a constant $\mathsf{get\_Challenge}$ to the passport that will answer by generating a nonce, *i.e.* a fresh random number.
2. Once the reader receives this nonce n_T, it will generate its own nonce n_R, as well as a key k_R that will be used later on to derive session keys. The reader encrypts the nonce n_T, its own nonce n_R as well as the

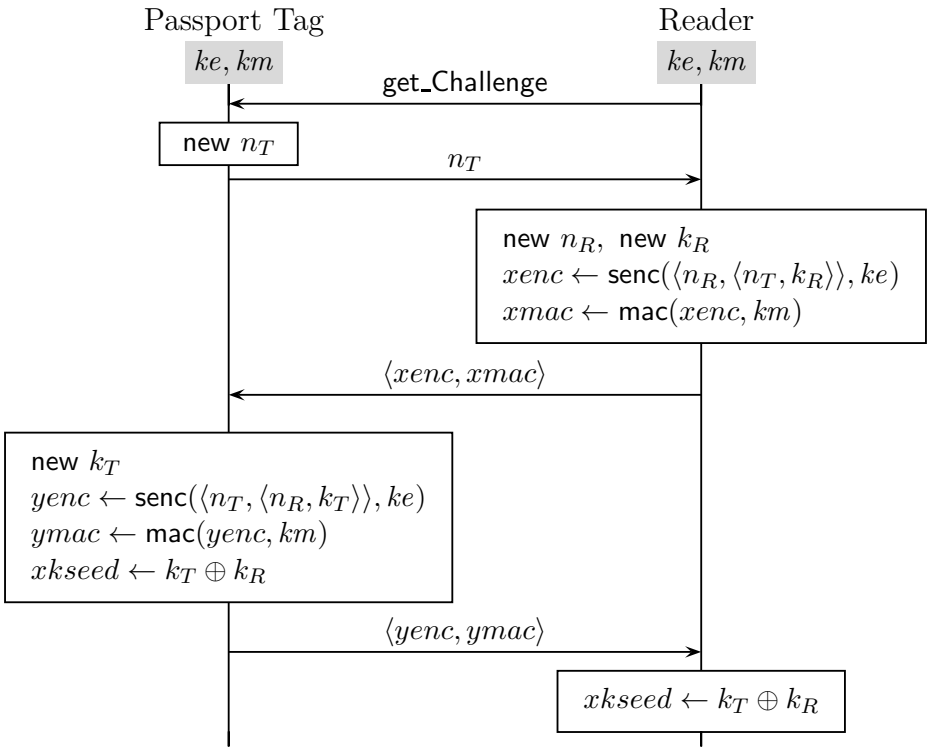

Figure 1: Basic Access Control protocol

key k_R with the (long-term) symmetric encryption key ke. This message $\mathsf{senc}(\langle n_R, \langle n_T, k_R\rangle\rangle, ke)$ is sent to the passport together with the associated MAC (with key km) to ensure that the encryption will be correctly transmitted to the passport.

3. The passport performs some checks. In particular, it checks that the MAC has been computed using the right key km, and that the nonce n_T it has generated at the first step of the protocol is inside the encryption. Once these checks have been performed, the passport sends to the reader a message similar to the one it has received using its own contribution k_T.
4. Again, the reader will perform the necessary checks before accepting the message, and in case of success two session keys will be generated from the value $xkseed$ obtained by applying the exclusive or operator

on k_R and k_T.

These two session keys are used to provide confidentiality, integrity, and authentication in subsequent communications. In particular, they are used to encrypt and MAC the messages exchanged during the execution of the Passive and Active Authentication protocols in order to ensure that only parties with physical access to the passport can read the data. The aim of establishing fresh session keys (instead of reusing ke and km at each session) is to make the passport *unlinkable*, a property that will be discussed in Section 2.3.

2.3. Some logical attacks on the BAC protocol

In this section, we describe two possible attacks. These attacks are purely logical in the sense that they do not require to break any cryptographic primitives.

Authentication issue. First, we would like to pinpoint the fact that the order in which the nonces n_R and n_T have been placed inside both encryptions is relevant. The careful reader will have noticed that the nonces have been swapped: the reader encrypts $\langle n_R, \langle n_T, k_R\rangle\rangle$ whereas the passport encrypts $\langle n_T, \langle n_R, k_T\rangle\rangle$. The purpose of this design is to avoid a *replay attack*. Indeed, without such a swap, a malicious user (who does not know the keys ke and km) would be able to simply replay the message he received from the reader without decrypting it and performing the checks. Such a message will be accepted by the reader and pass all checks performed by the reader. This means that the reader will end its session thinking (s)he has talked with the passport identified by ke and km, whereas this passport will not have really taken part in the protocol. Moreover, the key seed computed at the end will be $k_R \oplus k_R = 0$ and thus very different from the one that is supposed to be computed during a normal execution.

Unlinkability issue. Following the specification provided by the ICAO, each nation has implemented its own version. Unfortunately, as the specification is not completely comprehensive, each nation's passport has subtle differences. In particular, the standard specifies that the passport must reply with an error message to every ill formed or incorrect message from the reader, but it does not specify what the error message should be.

For example, in the French implementation, the passport tag replies different error messages depending on whether the nonce in $xenc$ is not n_T or

$xmac$ is not a correct MAC w.r.t. the key km [12]. An attacker who does not know the keys ke and km could then trace a passport in the following way:

1. He eavesdrops a session between an authentic reader and a passport P (with keys ke and km) and stores $m = \langle xenc, xmac \rangle$;
2. In a different session, he sends the message m and waits for the passport's answer;
3. Then, we distinguish two cases:
   (a) if he receives a nonce error then he knows that the passport succeeded to check the MAC and so this passport is P;
   (b) if he receives a MAC error then he knows that the passport is not the one with keys km (and ke), and therefore it is not P.

This attack makes it possible to detect when a particular passport comes into the range of a reader, which could be, for instance, placed by a doorway, in order to monitor when a target enters or leaves a particular building. To avoid the information leakage of these error messages, the specification should prescribe that, in case of failure, the passport yields the same message in both situations (as it is done for instance in e-passports from the UK).

We may note that in presence of honest participants who follow the protocol rules, the protocol works well, and the scenarios described above are not possible. However, it is important to ensure that these protocols work well in any situation, especially in the presence of malicious agents that may want to take advantage of the protocol and therefore do not necessarily follow the instructions specified by the protocols. Verifying cryptographic protocols in such a hostile environment is an essential feature which makes protocol verification a difficult task.

3. A variety of security properties

Cryptographic protocols aim at ensuring various security goals, depending on the application. The two most classical security properties are *secrecy* (also called *confidentiality*) and *authentication*.

Secrecy. This property concerns a message used by the protocol. This is typically a nonce or a secret key that should not become public. Even for this quite simple security property, several definitions have been proposed in the literature. When considering the notion of (weak) secrecy, a public message is a message that can be learnt by the attacker.

Authentication. Many security protocols aim at authenticating one agent to another: one agent should become sure of the identity of the other. There are also several variants of authentication. A taxonomy of these has been proposed by Lowe in [13].

Authentication and weak secrecy are both trace properties, that is, statements that something bad never occurs in any execution trace of a protocol. Several results and tools have been developed to analyse trace properties. However, privacy properties cannot be defined (or cannot be naturally defined) as trace properties. They are defined relying on a notion of indistinguishability. Intuitively, two protocols P and Q are indistinguishable if it is not possible for an attacker to decide whether (s)he is interacting with P or Q. This notion of indistinguishability is also used for defining a stronger notion of secrecy, and we may also rely on this notion of indistinguishability to compare a protocol with an idealised version of it. We will see in Section 4.4 how this notion of indistinguishability is formalised. Below, we simply list some security properties that can be formalised relying on such a notion.

Strong secrecy. This notion is stronger than (weak) secrecy, and related to the concept of indistinguishability. Intuitively, strong secrecy means that an adversary cannot see any difference when the value of the secret changes [14, 15, 16].

Anonymity. Frequently, communication between two principals reveals their identities and presence to third parties. Indeed, anonymity is in general not one of the explicit goals of common authentication protocols. However, we may want protocols that achieve this goal. It has been informally defined in the ISO/IEC 15408-2 standard as follows:

> [Anonymity] ensures that users may use a [protocol] without disclosing their identity.

It is usually formally defined (see [17, 18, 19]) as the fact that an observer cannot distinguish two scenarios where the same protocol is executed by different users.

Vote privacy. In the context of electronic voting, privacy means that the vote of a particular voter is not revealed to anyone. This is one of the fundamental security properties that an electronic voting system has to satisfy. Vote

privacy is typically defined (see *e.g.* [20, 21]) by the fact that an observer should not observe when two voters swap their votes, *i.e.* distinguish between a situation where Alice votes *yes* and Bob votes *no* and a situation where these two voters have voted the other way around.

In the context of electronic voting, some strong forms of vote-privacy are desirable too. For instance, *receipt-freeness* stipulates that a voter does not obtain any receipt information, which could be used by a coercer to prove that she voted in a certain way [22]. Receipt-freeness is intuitively a stronger property than privacy. Privacy says that an attacker cannot discern how a voter votes from any information that the voter necessarily reveals during the course of the election. Receipt-freeness says the same thing even if the voter voluntarily reveals additional information. Again, several formal definitions of such a property have already been proposed relying on the notion of indistinguishability (see *e.g.* [23, 24]). We may also be interested to ensure such a security property even if the strength of the encryption has eroded with the passage of time (which is unavoidable). This property, known as *everlasting privacy*, has again been formalised relying on the concept of indistinguishability [25].

Note that these concepts are not specific to the electronic voting applications and the definitions of privacy and receipt-freeness described above have also been reused and adapted to model privacy and receipt-freeness in *e.g.* on-line auction systems [26, 27].

Unlinkability. Protocols that keep the identity of their users secure may still allow an attacker to identify particular sessions as having involved the same principal. Such linkability attacks may, for instance, make it possible for a third party to trace the movements of someone carrying an RFID tag without him being able to notice anything (as the attack on the French version of the BAC protocol described in Section 2.3). Intuitively, protocols are said to provide unlinkability (or untraceability) according to the ISO/IEC 15408-2 standard, if they

> [...] ensure that a user may make multiple uses of [them] without others being able to link these uses together.

Formally, this is often defined as the fact that an attacker should not be able to distinguish a scenario in which the same agent (*i.e.,* the user) is involved in many sessions from one that involved different agents in each session. Following this intuition, slightly different definitions have been proposed (see

e.g. [17, 12, 28, 29]). A comparison between these definitions may be found in [30].

More generally, this notion allows one to express flexible notions of security by requiring indistinguishability between a protocol and an idealised version of it, that magically realises the desired properties. This is typically what is done to express security goals of a protocol in the Universal Composability (UC) framework. The universal composability paradigm has been quite successful in the computational approach [31]. The idea of UC is not, however, restricted to the computational setting, and has now a counterpart in symbolic models as well [32, 33].

4. Formalising protocols and properties

Several symbolic models have been proposed for cryptographic protocols. The first one has been described by Dolev and Yao [1] and several other models have been proposed since then. A unified model would enable better comparisons between the different existing results but unfortunately such a model does not exist currently. The reason for having several popular symbolic models probably comes from the fact that they have to achieve two antagonistic goals. On the one hand, models have to be as fine grained and expressive as possible to capture a large range of applications. One the other hand, models have to remain relatively simple in order to allow the design of verification procedures. In order to formally define the problems we are interested in, and to present the existing results, we will describe one such model which is intuitive enough. This model is inspired from cryptographic calculi, and actually pretty close to the applied-pi calculus [34].

4.1. Messages

In symbolic models, messages are a key concept. Whereas messages are bit-strings in the real-world (and in the computational approach as well), they are modelled using first-order terms within the symbolic model. Atoms can be for instance nonces, keys, or agent identities. Examples of function symbols are concatenation, asymmetric and symmetric encryptions or digital signatures. We consider an infinite set $\mathcal{N}$ of *names* which are used to represent keys and nonces (*e.g.* k, n); and two infinite and disjoint sets of *variables*, denoted $\mathcal{X}$ and $\mathcal{W}$. Variables in $\mathcal{X}$ will typically be used to refer to unknown parts of messages expected by participants and will be denoted x,

y, z, ... Variables in $\mathcal{W}$ will be used to store messages learnt by the attacker and will be denoted w, w_1, w_2, ... We assume a signature Σ, *i.e.* a set of function symbols together with their arity. Given a signature $\mathcal{F}$ and a set of initial data A, we denote by $\mathcal{T}(\mathcal{F}, \mathsf{A})$ the set of *terms* built from elements of A by applying function symbols in $\mathcal{F}$. Given a term u, we note $vars(u)$ the variables that occur in u. A message is a *ground* term, *i.e.* a term that does not contain any variable. Some works rely on a sort system for terms, and consider that only atomic data may occur at a key position (atomic keys).

Example 1. *Consider the signature*

$$\Sigma = \{\mathsf{senc}, \mathsf{sdec}, \langle\,\rangle, \mathsf{proj}_1, \mathsf{proj}_2, \mathsf{mac}, \oplus, 0\}.$$

We use the binary symbols senc *and* sdec *to represent symmetric encryption and decryption. Pairing is modelled using the binary symbol* $\langle\,\rangle$*, whereas projections are modelled using the unary symbols* proj_1 *and* proj_2*. The binary function symbol* $\oplus$ *and the constant* 0 *are used to model the exclusive or operator, and the binary symbol* mac *is used to model message authentication code.*

There are two different ways to assign a meaning to function symbols, which we describe next.

Equational theory. To give a meaning to these function symbols, we associate an equational theory E to the signature Σ. An equational theory is a Σ-congruence on terms that is closed under substitutions of terms for variables. We usually require the equational theory to be closed under one-to-one renaming (of names in $\mathcal{N}$), but not necessarily closed under substitutions of arbitrary terms for names. Usually, an equational theory is generated from a finite set of equations $M = N$ with $M, N \in \mathcal{T}(\Sigma, \mathcal{X})$. In this case, we have that E is closed by substitutions of terms for names.

Example 2. *To reflect the algebraic properties of the exclusive or operator, as well as the encryption/decryption and pairing/projection functions, we may consider the equational theory generated by the following set of equations* ($i \in \{1, 2\}$)*:*

$$\begin{array}{rcl@{\qquad}rcl} \mathsf{proj}_i(\langle x_1, x_2\rangle) & = & x_i & \mathsf{sdec}(\mathsf{senc}(x, y), y) & = & x \\ x \oplus 0 & = & x & (x \oplus y) \oplus z & = & x \oplus (y \oplus z) \\ x \oplus x & = & 0 & (x \oplus y) & = & (y \oplus x) \end{array}$$

In such a case, we have that $\mathsf{senc}(a \oplus (b \oplus a), k) =_{\mathsf{E}} \mathsf{senc}(b, k)$.

Rewriting system. Some frameworks rely on a rewriting system to give a meaning to function symbols. In such a case, function symbols in Σ are split into constructor/destructor symbols, namely $\Sigma = \Sigma_c \uplus \Sigma_d$, and a rewriting system is used to reduce destructor symbols. For instance, assuming the signature given in Example 1, let us consider $\Sigma_c = \{\mathsf{senc}, \langle\ \rangle, \mathsf{mac}, \oplus, 0\}$ and $\Sigma_d = \{\mathsf{sdec}, \mathsf{proj}_1, \mathsf{proj}_2\}$ together with the three following rewriting rules:

$$\mathsf{proj}_i(\langle x_1, x_2\rangle) \to x_i \text{ with } i \in \{1, 2\}, \text{ and } \mathsf{sdec}(\mathsf{senc}(x, y), y) \to x.$$

This rewriting system allows us to rewrite terms until reaching a message (terms that only use constructor symbols). In case such a message cannot be reached, we say that the *computation failed*.

This gives us two slightly different ways to model *e.g.* symmetric encryption, with some fundamental differences. Relying on a modelling using equations, an attacker will be able to apply the decryption algorithm using a key k on top of a term which is not a cipher-text. Considering a modelling of encryption/decryption using a rewriting rule, such a computation will fail, and therefore the attacker will be able to see whether the message is a cipher-text (encrypted with the expected key) or not. Some frameworks, such as the one used in the ProVerif tool [35], allow both kinds of function symbols: some are equationally defined whereas some other are defined through rewriting rules.

For the sake of clarity, we will assume that all function symbols are given a meaning through an equational theory only. Considering rewriting systems would require some adaptation (for instance, the fact that a computation fails is something that can be observed by the attacker).

Relying on equational theories gives us enough flexibility to model a variety of cryptographic primitives. For instance, it is possible to model a *blind signature* scheme (a primitive that is often used in e-voting protocols) as follows [20]:

$$\mathsf{unblind}(\mathsf{blind}(x, y), y) = x \qquad \mathsf{checksign}(\mathsf{sign}(x, y), \mathsf{pk}(y)) = x$$
$$\mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x, y), z), y) = \mathsf{sign}(x, z)$$

Intuitively, an agent can apply the function blind on a message using a blinding factor of his choice. Then, it is possible to retrieve the original message only for one who knows this blinding factor. Note that the last equation also permits one to extract a signature out of a blind signature, but only when the blinding factor is known.

4.2. Assembling terms into frames

At a particular point in time, while engaging in one or more sessions of one or more protocols, an attacker may know a sequence of messages (ground terms) $u_1, \ldots, u_\ell$. This means that he knows all messages and also their order. So it is not enough for us to say that the attacker knows the set of terms $\{u_1, \ldots, u_\ell\}$. In the applied-pi calculus [34], such a sequence of messages is organised into a *frame*, *i.e.* a substitution of the form:

$$\phi = \{w_1 \mapsto u_1, \ldots, w_\ell \mapsto u_\ell\}.$$

The variables $w_1, \ldots, w_\ell$ from $\mathcal{W}$ enable us to refer to each message u_i, and these variables will allow us to make explicit the order in which these messages are sent.

For modelling purposes, we split the signature Σ into two parts, Σ_{pub} and Σ_{priv} (this is orthogonal to the splitting that may have been done between constructors and destructors mentioned previously). An attacker builds his own messages by applying public function symbols (*i.e.,* in Σ_{pub}) to ground terms he already knows and that are available through variables from $\mathcal{W}$. Formally, a computation done by the attacker is a *recipe*, *i.e.* a term in $\mathcal{T}(\Sigma_{\mathsf{pub}}, \mathcal{W})$. The application of a substitution σ to a term u is written $u\sigma$, and we denote by $dom(\sigma)$ its *domain*. For two recipes R, R' and a frame ϕ, we note $(R =_{\mathsf{E}} R')\phi$ when $R\phi =_{\mathsf{E}} R'\phi$.

Example 3. *Consider the signature defined in Example 1 together with the equational theory presented in Example 2. Let ϕ be the following frame:*

$$\phi = \{w_1 \mapsto \mathsf{senc}(n_1, k),\ w_2 \mapsto \mathsf{senc}(n_2, k),\ w_3 \mapsto \mathsf{senc}(n_3, k),\ w_4 \mapsto k\}$$

We have that $R_1 = \mathsf{sdec}(w_1, w_4)$, $R_2 = \mathsf{sdec}(w_2, w_4)$, and $R_3 = \mathsf{sdec}(w_3, w_4)$ are three recipes. These recipes allow the attacker to compute the messages $R_1\phi$, $R_2\phi$, and $R_3\phi$. These terms are equal modulo E to the names n_1, n_2, and n_3.

Several notions of equivalence between processes have been introduced in the literature to express indistinguishability, but actually they all rely on the notion of *static equivalence*. Intuitively, an attacker can distinguish two frames if he is able to perform a test that succeeds in one frame, whereas it fails in the other. More formally, we have that:

Definition 1. *Two frames ϕ and ϕ' are in* static equivalence*, written $\phi \sim_{\mathsf{E}} \phi'$, when $dom(\phi) = dom(\phi')$, and for any recipes $R_1, R_2 \in \mathcal{T}(\Sigma_{\mathsf{pub}}, dom(\phi))$, we have that:*

$$(R_1 =_{\mathsf{E}} R_2)\phi \text{ if, and only if, } (R_1 =_{\mathsf{E}} R_2)\phi'.$$

Example 4. *Resuming Example 3, consider the frame:*

$$\phi' = \{w_1 \mapsto \mathsf{senc}(n'_1, k'),\ w_2 \mapsto \mathsf{senc}(n'_2, k'),\ w_3 \mapsto \mathsf{senc}(n'_1 \oplus n'_2, k'),\ w_4 \mapsto k'\}.$$

We have that $R \stackrel{\mathsf{def}}{=} R_1 \oplus R_2$ and $R' \stackrel{\mathsf{def}}{=} R_3$ (where R_1, R_2, and R_3 are as defined in Example 3) are two recipes such that $(R =_{\mathsf{E}} R')\phi'$ whereas this equality does not hold in ϕ. Therefore the frames ϕ and ϕ' are not *in static equivalence.*

Consider the frame ψ (resp. ψ') obtained by removing its last element $w_4 \mapsto k$ (resp. $w_4 \mapsto k'$) from ϕ (resp. ϕ'). We have that ψ and ψ' are in static equivalence.

Many decidability and complexity results for static equivalence already exist (*e.g.* [36, 37, 38]), and some of these procedures have even been implemented (*e.g.* KISS [39], YAPA [40], FAST [41]). These results cover a wide class of cryptographic primitives as long as they are modelled through convergent equational theories (*i.e.,* theories in which equations can be oriented and form a convergent rewriting system).

However, static equivalence is a static notion, and does not take into account the dynamic behaviour of the underlying protocols. Static equivalence represents a passive attacker who can only observe messages that are sent on the public network, and is not powerful enough to mount the attacks described in Section 2.3. Even if this notion of static equivalence plays an important role for the analysis of security protocols in the presence of an active attacker (*i.e.,* an attacker who may interact and interfere with the protocol), it remains challenging to obtain decidability results for the active case, especially in presence of algebraic properties. The results that have already been obtained in this direction (and the associated tools) are described in Section 5.

4.3. Protocols

We assume an infinite set $\mathcal{C}h = \mathcal{C}h_0 \uplus \mathcal{C}h^{\mathsf{fresh}}$ of *channel names*, where $\mathcal{C}h_0$ and $\mathcal{C}h^{\mathsf{fresh}}$ are infinite and disjoint. Intuitively, channels of $\mathcal{C}h^{\mathsf{fresh}}$ will be used to instantiate channels when they are generated during the execution of the protocol. They should not be part of a protocol specification.

Syntax. Protocols are modelled through *processes* built by the grammar given below (where $c, c' \in \mathcal{C}h_0$, $x \in \mathcal{X}$, $n \in \mathcal{N}$, and, $u, u_1, u_2 \in \mathcal{T}(\Sigma, \mathcal{N} \cup \mathcal{X})$):

$$
\begin{array}{rcll}
P, Q & := & 0 & \text{null} \\
& & \mid P \mid Q & \text{parallel} \\
& & \mid \mathsf{in}(c, x).P & \text{input} \\
& & \mid \mathsf{out}(c, u).P & \text{output} \\
& & \mid\ !P & \text{replication} \\
& & \mid \mathsf{new}\ n.P & \text{restriction} \\
& & \mid \mathsf{new}\ c'.\mathsf{out}(c, c').P & \text{creation of public channel} \\
& & \mid \mathsf{if}\ u_1 = u_2\ \mathsf{then}\ P\ \mathsf{else}\ Q & \text{conditional}
\end{array}
$$

The process 0 denotes the null process that does nothing. The process $P \mid Q$ runs P and Q in parallel. The process $\mathsf{in}(c, x).P$ waits to receive a message on the public channel c, and then continues as P but with x replaced by the received message. The process $\mathsf{out}(c, u).P$ outputs a term u on the channel c, and then continues as P. The process $!P$ executes an infinite number of copies of P in parallel. The restriction $\mathsf{new}\, n.P$ is used to model the creation in a process of new random numbers (*e.g.*, nonces or key material). The process $\mathsf{new}\, c'.\mathsf{out}(c, c').P$ is a special construction for creating new channels: any new channel should be made public immediately. Intuitively, we consider here only public channels. These fresh channel names are used to identify a process, similarly to a session identifier for example. The process $\mathtt{if}\ u_1 = u_2\ \mathtt{then}\ P\ \mathtt{else}\ Q$ runs as P if the terms u_1 and u_2 are equal in the equational theory, and as Q otherwise. Note that the terms u, u_1, and u_2 that occur in the grammar may contain variables but the terms will become ground when the evaluation will take place. Note also that $\mathsf{new}\, n.P$ and $\mathsf{in}(c, x).P$ are binding constructs, respectively for the name n and for the variable x, and in both cases the scope of the binding is P.

We consider only a fragment of the applied-pi calculus. In particular, we do not allow channel passing nor internal communication. Such a calculus will be (almost) sufficient to present all the existing results. A *protocol* is a ground process, *i.e.* a process whose variables are in the scope of an input.

For the sake of clarity, we often omit the null process, and we omit the else branch of a conditional when it contains the 0 process.

Example 5. *We are now able to model the French variant of the tag's role of the BAC protocol (see Figure 1) as a process parameterised by two (long-*

term) keys ke, and km.

$$\begin{array}{rcl} P_{\mathrm{Tag}}(ke, km) & := & \mathsf{in}(c_T, z).\mathsf{new}\ n_T.\mathsf{out}(c_T, n_T).\mathsf{in}(c_T, x). \\ & & \mathsf{if}\ \mathsf{mac}(\pi_1(x), km) = \pi_2(x) \\ & & \mathsf{then\ if}\ \pi_1(\pi_2(\mathsf{sdec}(\pi_1(x), ke))) = n_T \\ & & \quad \mathsf{then\ new}\ k_T.\mathsf{out}(c_T, \langle m, \mathsf{mac}(m, km)\rangle) \\ & & \quad \mathsf{else\ out}(\mathsf{error}_{\mathsf{Nonce}}) \\ & & \mathsf{else\ out}(\mathsf{error}_{\mathsf{Mac}}) \end{array}$$

where $m = \mathsf{senc}(\langle n_T, \langle \pi_1(\mathsf{sdec}(\pi_1(x), ke)), k_T\rangle\rangle, ke)$. *If* $P_{\mathrm{Reader}}(ke, km)$ *is the process modelling the reader, the protocol BAC with many readers and tags that can play arbitrary many sessions can be modelled through the following process:*

$$P_{\mathrm{BAC}} = !\ \mathsf{new}\ ke.\mathsf{new}\ km.\ !\ (P_{\mathrm{Tag}}(ke, km) \mid P_{\mathrm{Reader}}(ke, km)).$$

Configurations represent processes having already evolved by *e.g.* disclosing some terms to the environment.

Definition 2. *A* configuration *is a pair* $(\mathcal{P}; \phi)$ *where* $\mathcal{P}$ *is a multiset of* ground processes*; and* $\phi = \{w_1 \mapsto u_1, \ldots, w_n \mapsto u_n\}$ *is a* frame.

We implicitly assume that null processes are removed from a configuration. The applied-pi calculus as introduced in [34] does not introduce this notion of configuration but considers instead the notion of *extended processes* together with a notion of structural equivalence to identify processes that are identical up to some rearrangement of their structure (*e.g.* $P \mid Q$ and $Q \mid P$). A configuration can be seen as a more canonical way to represent an extended process, and this will avoid us to rely on a notion of structural equivalence.

A configuration is said to be *initial* when it does not use channel names from $\mathcal{C}h^{\mathsf{fresh}}$.

In some cases, it is helpful to know which process has executed a given observable action. Hence, some methods and tools we will discuss in Section 5 consider the class of the *simple processes*.

Definition 3. *A* simple process *is a process of the form:*

$$!\mathsf{new}\ c_1.\mathsf{out}(c'_1, c_1).B_1 \mid \cdots \mid !\mathsf{new}\ c_n.\mathsf{out}(c'_n, c_n).B_n \mid B_{n+1} \mid \cdots \mid B_{n+k}$$

where all c_i, c'_i *are distinct and for any* $1 \leq i \leq n + k$, B_i *is a* basic process *built on channel* c_i*, where a* basic process *on channel* c *is a process built using the following grammar:*

$$B, B' := 0 \mid \mathsf{in}(c, x).B \mid \mathsf{out}(c, u).B \mid \mathsf{new}\, n.B \mid \mathsf{if}\ u_1 = u_2\ \mathsf{then}\ P\ \mathsf{else}\ Q$$

This class is reasonable given that the attacker often knows with whom he is communicating. For simple processes, this is reflected by the fact that concurrent processes use different channels that are observable by the attacker.

Semantics. The semantics is given by a labelled transition system (LTS) on configurations (see Figure 2). This labelled operational semantics allows one to avoid the quantification over all contexts when analysing a protocol in presence of an arbitrary attacker.

The rules are quite standard and correspond to the intuitive meaning of the syntax given in the previous section. When a process emits a message, we distinguish two cases. The rule Out corresponds to the output of a term by some process: the corresponding term is added to the frame of the current configuration, which means that the attacker can now access the sent term. The rule Ch corresponds to the special case of an output of a freshly generated channel name. In such a case, the channel is not added to the frame but it is implicitly assumed known to the attacker, as all the channel names.

These rules define the relation $\xrightarrow{\ell}$, where ℓ is either an input, an output, or a silent action τ. The relation $\xrightarrow{\mathsf{tr}}$ where tr denotes a sequence of labels is defined in the usual way, whereas the relation $\stackrel{\mathsf{tr}'}{\Longrightarrow}$ on configurations is defined by: $K \stackrel{\mathsf{tr}'}{\Longrightarrow} K'$ if, and only if, either $K = K'$ and $\mathsf{tr}' = \epsilon$ (the empty trace); or there exists a sequence tr such that $K \xrightarrow{\mathsf{tr}} K'$ and tr' is obtained by erasing all occurrences of the silent action τ in tr.

Given an initial configuration $K_0 = (\mathcal{P}; \phi)$, we define its set of traces as follows:

$$\mathsf{traces}(K_0) = \{(\mathsf{tr}, \phi') \mid K_0 \stackrel{\mathsf{tr}}{\Longrightarrow} (\mathcal{P}'; \phi') \text{ for some configuration } (\mathcal{P}'; \phi')\}.$$

Example 6. *Continuing Example 5, we consider the following configuration*

$$K_A = (\{P_{\mathrm{Tag}}(ke^A, km^A)\}; \{w_0 \mapsto \langle m_0, \mathsf{mac}(m_0, km^A)\rangle\})$$

where $m_0 = \mathsf{senc}(\langle n_R^0, \langle n_T^0, k_R^0\rangle\rangle, ke^A)$. *This configuration represents a scenario where the attacker has eavesdropped a message* $\langle m_0, \mathsf{mac}(m_0, km^A)\rangle$ *from a previous session between a reader and Alice's tag with keys* ke^A, *and* km^A; *and now the attacker is again in presence of Alice's tag. We have that* $K_A \stackrel{\mathsf{tr}_A}{\Longrightarrow} (\emptyset; \phi_A)$ *where*

THEN
$(\{\texttt{if } u_1 = u_2 \texttt{ then } P_1 \texttt{ else } P_2\} \uplus \mathcal{P}; \phi) \xrightarrow{\tau} (P_1 \uplus \mathcal{P}; \phi)$ when $u_1 =_{\mathsf{E}} u_2$

ELSE
$(\{\texttt{if } u_1 = u_2 \texttt{ then } P_1 \texttt{ else } P_2\} \uplus \mathcal{P}; \phi) \xrightarrow{\tau} (P_2 \uplus \mathcal{P}; \phi)$ when $u_1 \neq_{\mathsf{E}} u_2$

IN
$(\{\mathsf{in}(c, z).P\} \uplus \mathcal{P}; \phi) \xrightarrow{\mathsf{in}(c,R)} (P\{z \mapsto R\phi\} \uplus \mathcal{P}; \phi)$ where $vars(R) \subseteq dom(\phi)$

OUT
$(\{\mathsf{out}(c, u).P\} \uplus \mathcal{P}; \phi) \xrightarrow{\mathsf{out}(c,w)} (P \uplus \mathcal{P}; \phi \cup \{w \mapsto u\})$ where $w \in \mathcal{W}$ is fresh

CH
$(\{\mathsf{new}\ c'.\mathsf{out}(c, c').P\} \uplus \mathcal{P}; \phi) \xrightarrow{\mathsf{outCh}(c,ch)} (P\{c' \mapsto ch\} \uplus \mathcal{P}; \phi)$
where $ch \in \mathcal{C}h^{\mathsf{fresh}}$ is fresh

NEW $(\{\mathsf{new}\ n.P\} \uplus \mathcal{P}; \phi) \xrightarrow{\tau} (P\{n \mapsto n'\} \uplus \mathcal{P}; \phi)$ where $n' \in \mathcal{N}$ is fresh

PAR $(\{P_1 \mid P_2\} \uplus \mathcal{P}; \phi) \xrightarrow{\tau} (\{P_1, P_2\} \uplus \mathcal{P}; \phi)$

REPL $(\{!P\} \uplus \mathcal{P}; \phi) \xrightarrow{\tau} (\{!P, P\} \uplus \mathcal{P}; \phi)$

Figure 2: Semantics

- $\mathsf{tr}_A = \mathsf{in}(c_T, \mathsf{get\_Challenge}).\mathsf{out}(c_T, w_1).\mathsf{in}(c_T, w_0).\mathsf{out}(c_T, w_2)$*; and*
- $\phi_A = \{w_0 \mapsto \langle m_0, \mathsf{mac}(m_0, km^A)\rangle;\ w_1 \mapsto \langle m_A, \mathsf{mac}(m_A, km^A)\rangle;\ w_2 \mapsto \mathtt{error}_{\mathsf{Nonce}}\}$ *(for some m_A).*

Our calculus has similarities with the spi calculus [42]. The key difference concerns the way in which cryptographic primitives are handled. The spi calculus has a fixed set of built-in primitives (namely, symmetric and public key encryption), while our calculus allows a wide variety of primitives to be defined by means of an equational theory as in the applied-pi calculus. Process algebras are not the only way to model protocols. We may at least mention the multiset rewriting (MSR) model that has been introduced in [43] to model reachability properties (*e.g.* weak secrecy, authentication), and the strand space model [44] that comes with an appealing graphical representation, and some proof techniques. These two models have been recently extended to capture an equivalence based property similar to the notion of diff-equivalence that we will introduce in the following section.

4.4. Equivalences

In order to express the security properties introduced in Section 3, we need to formally define the notion of indistinguishability we are interested in. Intuitively, two processes are indistinguishable if an attacker has no way to tell them apart. A natural starting point is to say that processes P and Q are indistinguishable if they can output on the same channels, no matter the context in which they are placed. The quantification over contexts makes this definition hard to use in practice. Therefore indistinguishability notions, which are more suitable for both manual and automatic reasoning, have been proposed. All these notions rely on a labelled transition semantics as the one presented in Section 4.3 to reason about protocols that may interact with an environment that models an arbitrary attacker.

Here, we make the choice to directly present a labelled semantics together with equivalence notions that are based on this semantics and therefore avoid the quantification over contexts required when using a reduction semantics. Actually linking these two semantics and their associated notions of equivalence is not an easy task. Starting with the pioneering work of Milner and Sangiorgi [45], this problem has been addressed for different calculi and different notions of equivalence in several papers (*e.g.* pi-calculus, spi-calculus [46, 47], applied-pi calculus [34], and psi-calculus [48]).

Trace equivalence. The notion that seems to be the most appropriate to capture the notion of indistinguishability we are interested in is the notion of trace equivalence.

Definition 4. *Let K_P and K_Q be two initial configurations, $K_P \sqsubseteq_t K_Q$ if for every $(\mathsf{tr}, \phi) \in \mathsf{traces}(K_P)$, there exists $(\mathsf{tr}', \phi') \in \mathsf{traces}(K_Q)$ such that $\mathsf{tr} = \mathsf{tr}'$ and $\phi \sim \phi'$. We say that K_P and K_Q are* trace equivalent*, denoted by $K_P \approx_t K_Q$, if $K_P \sqsubseteq_t K_Q$ and $K_Q \sqsubseteq_t K_P$.*

Example 7. *In order to formalise whether the attacker is able to distinguish between Alice's tag and Bob's tag, one may want to check if K_A is trace equivalent to*

$$K_B = (\{P_{\text{Tag}}(ke^B, km^B)\}; \{w_0 \mapsto \langle m_0, \mathsf{mac}(m, km^A)\rangle\})$$

This equivalence actually fails to hold. We have that $(\mathsf{tr}_B, \phi_B) \in \mathsf{traces}(K_B)$ for some trace tr_B that has exactly the same observable actions as tr_A. However, the only possible resulting frame is ϕ_B (for some message m_B):

$$\{w_0 \mapsto \langle m_0, \mathsf{mac}(m_0, km^A)\rangle;\ w_1 \mapsto \langle m_B, \mathsf{mac}(m_B, km^B)\rangle;\ w_2 \mapsto \mathtt{error}_{\mathsf{Mac}}\}$$

It is easy to see that $\phi_A \sim_{\mathsf{E}} \phi_B$ *does* not *hold. Indeed, using recipes* $R_1 = w_2$ *and* $R_2 = \mathtt{error}_{\mathsf{Nonce}}$*, we have that* $(R_1 =_{\mathsf{E}} R_2)\phi_A$ *but* $(R_1 \neq_{\mathsf{E}} R_2)\phi_B$*.*

Hence, just by looking at the second output of the tag and checking whether it is equal to the public constant $\mathtt{error}_{\mathsf{Nonce}}$*, the attacker is able to learn if he was interacting with Bob or Alice. This formalises the unlinkability attack discussed in Section 2.2 for the specific case of two sessions.*

Note that, in case the two error messages $\mathtt{error}_{\mathsf{Nonce}}$ *and* $\mathtt{error}_{\mathsf{Mac}}$ *were equal as in the UK version, one would have* $K_A^{\mathsf{UK}} \approx_t K_B^{\mathsf{UK}}$*. This is a non-trivial equivalence that can be established using* e.g *the* APTE *tool presented in Section 5.*

Labelled bisimilarity. Showing trace equivalence properties is a very difficult task. The notion of labelled bisimilarity for the spi-calculus has been introduced to approximate trace equivalence [42]. The fact that labelled bisimilarity is based on a notion of step-by-step simulation between processes makes this notion sometimes easier to establish directly.

Definition 5. Labelled bisimilarity $\approx$ *is the largest symmetric relation* $\mathcal{R}$ *on configurations such that* $(\mathcal{P};\phi)\ \mathcal{R}\ (\mathcal{Q};\psi)$ *implies:*

1. *static equivalence:* $\phi \sim \psi$*;*
2. *if* $(\mathcal{P};\phi) \xrightarrow{\tau} K_P$*, then* $(\mathcal{Q};\psi) \stackrel{\tau}{\Rightarrow} K_Q$ *and* $K_P\ \mathcal{R}\ K_Q$ *for some* K_Q*;*
3. *if* $(\mathcal{P};\phi) \xrightarrow{\alpha} K_P$*, then* $(\mathcal{Q};\psi) \stackrel{\alpha}{\Rightarrow} K_Q$ *and* $K_P\ \mathcal{R}\ K_Q$ *for some* K_Q*.*

Two initial configurations K_P *and* K_Q *are labelled bisimilar if* $K_P \approx K_Q$*.*

It is well-known that labelled bisimilarity implies trace equivalence whereas the converse is false in general. Actually, it has been proved in [49] that these two notions coincide for a large class of processes that include in particular the class of simple processes as described in Definition 3.

Diff-equivalence. Another notion of equivalence that has been extensively used in the context of cryptographic protocols verification is the notion of *diff-equivalence*. Such a notion is defined on bi-processes that are pairs of processes that have the same structure and differ only in the choice of terms they use. The syntax is similar to the one introduced in Section 4.3 but each term u has to be replaced by a bi-term written $\mathsf{choice}[u_1, u_2]$ (using PROVERIF syntax). Given a bi-process P, the process $\mathsf{fst}(P)$ is obtained

by replacing all occurrences of $\mathsf{choice}[u_1, u_2]$ with u_1. Similarly, $\mathsf{snd}(P)$ is obtained by replacing $\mathsf{choice}[u_1, u_2]$ with u_2. These notations are also used for bi-frames.

The semantics of bi-processes is defined as expected via a relation that expresses when and how a bi-configuration may evolve. A bi-process reduces if, and only if, both sides of the bi-process reduce in the same way: *e.g.* a conditional has to be evaluated in the same way on both sides. For instance, the Then and Else rules are as follows:

Then
$$(\{\texttt{if } \mathsf{choice}[u_1, u_2] = \mathsf{choice}[v_1, v_2] \texttt{ then } Q_1 \texttt{ else } Q_2\} \uplus \mathcal{P}; \phi) \xrightarrow{\tau}_{\mathsf{bi}} (Q_1 \uplus \mathcal{P}; \phi) \quad \text{when } u_1 =_{\mathsf{E}} v_1 \text{ and } u_2 =_{\mathsf{E}} v_2$$

Else
$$(\{\texttt{if } \mathsf{choice}[u_1, u_2] = \mathsf{choice}[v_1, v_2] \texttt{ then } Q_1 \texttt{ else } Q_2\} \uplus \mathcal{P}; \phi) \xrightarrow{\tau}_{\mathsf{bi}} (Q_2 \uplus \mathcal{P}; \phi) \quad \text{when } u_1 \neq_{\mathsf{E}} v_1 \text{ and } u_2 \neq_{\mathsf{E}} v_2$$

When the two sides of the bi-process reduce in different ways, the bi-process blocks. The relation $\overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}}$ on bi-processes is therefore defined as for processes. This leads us to the following notion of diff-equivalence.

Definition 6. *An initial bi-configuration K_0 satisfies* diff-equivalence *if for every bi-configuration $K = (\mathcal{P}; \phi)$ such that $K_0 \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} K$ for some trace tr, we have that:*

- $\mathsf{fst}(\phi) \sim \mathsf{snd}(\phi)$*;*
- *if $\mathsf{fst}(K) \xrightarrow{\alpha} A_L$ then there exists a bi-configuration K' such that $K \xrightarrow{\alpha}_{\mathsf{bi}} K'$ and $\mathsf{fst}(K') = A_L$ (and similarly for snd).*

As expected, this notion of diff-equivalence is actually stronger than the usual notion of labelled bisimilarity, and thus trace equivalence. It may be the case that the two sides of the bi-process reduce in different ways (e.g. taking two different branches in a conditional) but still produce the same observable actions. This strong notion of diff-equivalence happens to be sufficient to establish some interesting equivalence-based properties such as strong secrecy, and anonymity. However, this notion is actually too strong to establish for example vote privacy for many interesting e-voting protocols [21], or unlinkability as defined in [12].

For instance, looking back to Example 7 (when error messages are equal), it can be shown that $K_A^{\mathsf{UK}} \approx_t K_B^{\mathsf{UK}}$. On the other hand, K_A^{UK} and K_B^{UK} are

not related by diff-equivalence. Indeed, the first three observable actions of $\mathsf{tr}_A/\mathsf{tr}_B$ are executable, but this results in a bi-process with a conditional that evaluates differently on both sides. Therefore, even if the error message outputted on both sides is the same, diff-equivalence does not hold.

5. Methods and tools for verifying equivalence-based properties

Modelling protocols using the symbolic approach allows one to benefit from machine support through the use of various existing techniques, ranging from model-checking to resolution and rewriting techniques. Aiming at machine support is really relevant since manual proofs are error-prone, tedious and hardly verifiable. Moreover, new protocols are developed quite frequently and need to be verified quickly. Nevertheless, verifying a security property in such a setting (and especially those expressed using the notion of equivalence) remains a difficult problem which is actually undecidable [50, 51].

5.1. Bounded number of sessions

In order to design decision procedures, a reasonable assumption is to bound the number of protocol sessions (*i.e.,* forbid replication), thereby limiting the length of execution traces. Under such an assumption, the first decision procedure towards automatic verification of equivalence between protocols dates back to [50], where a fragment of the spi calculus (no replication, no else branch) is considered. Note that, even under this assumption, infinitely many traces remain, since each input may be fed infinitely many different messages. This issue has been tackled in various ways using forms of symbolic execution and the development of dedicated procedures. Obtaining a symbolic semantics to avoid potentially infinite branching of execution trees due to inputs from the environment is often a first step towards automation of equivalence. Depending on the expressivity of the calculus and the way its semantics is given, this task can be quite cumbersome (*e.g.* applied-pi calculus [52], spi calculus [53, 54], psi calculus [55]) and sometimes only leads to incomplete procedures.

A table summarising the main features of existing tools dedicated to bounded verification is given in Table 1.

5.1.1. Constraint solving approaches

Baudet targets the decision of security of protocols against off-line guessing attacks defined using static equivalence between open frames (*i.e.,* frames

with some unknown parts constrained with some deducibility and equality constraints) [56]. The main novelty of his work was to design a constraint solving procedure that is not only able to solve satisfiability problems (sufficient for reachability properties) but also to establish equivalences (*i.e.,* two systems have the same sets of solutions), which are needed when one wants to verify equivalence-based security properties. This is done for a user-defined equational theory given in the form of a subterm convergent rewriting system (*i.e.,* convergent and such that the right-hand side of each rewriting rule is actually a syntactic subterm of the left-hand side). As a result, this work allows for verifying trace equivalence of simple processes (with no else branch) for all the standard primitives [49].

A shorter proof of the result by Baudet is given in [57]. It is shown that if two processes are not equivalent, then there must exist a small witness of non-equivalence, and a decision procedure can be derived by checking every possible small witness. The main issue with all the results mentioned so far is practicality. Consequently, they have not been implemented.

A decision procedure for a stronger notion of trace equivalence (namely *open bisimulation*) has been proposed in [58] and implemented in the tool SPEC[2]. The procedure deals with a fixed set of cryptographic primitives, namely symmetric encryption and pairs, and protocols with no else branch. The procedure is sound and complete w.r.t. open bisimulation (a notion that is strictly stronger than trace equivalence [59]) and its termination is proved. The attacker's deductive ability is modelled as logical rules in sequent calculus, and procedures deciding message deduction and message indistinguishability are defined as proof-search strategies. Finally, the proposed procedure iteratively builds an open bisimulation from the two initial processes by symbolically executing them and checks that possible instantiations are coherent on both sides.

For a fixed but richer set of cryptographic primitives (*i.e.,* symmetric/asymmetric encryptions, signature, pair, and hash functions), a different procedure, presented in [60] (improved version of [61]), allows to decide equivalence of two *sets* of constraint systems that may also feature *disequality tests*. Dealing with disequality tests and sets of constraint systems is needed in the presence of protocols with else branches (different symbolic executions

[2]http://www.ntu.edu.sg/home/atiu/spec/index.html

	APTE [62]	AKISS [65]	SPEC [58]
Equivalence	$\approx_t$	$\approx_{ct}$, $\approx_{ft}$	open bisim.
Primitives	standard	convergent with finite variant	pair & sym. encryption
Class of protocols	full	linear role with equality tests	linear role with filtering
Input syntax	applied-pi calculus		spi calculus
Termination	proved	proved for sub. convergent	proved
Exploration	forward		

Table 1: Main features of existing tools (for a bounded number of sessions)

may be associated to a single symbolic trace). Actually, the procedure presented in [60] allows for slightly more general processes than those presented in Section 4 since it deals with private channels and internal communications. The tool APTE [62] implements the procedure described in [60]. This procedure explores all possible symbolic traces and computes all possible resulting symbolic constraint systems on both sides. This forward symbolic exploration of two processes is finite since all symbolic traces have a bounded length and the exploration is finitely branching since inputs are abstracted away by variables and constraints. The procedure then checks the symbolic equivalence of all the resulting pairs of sets of constraint systems. Recently, this procedure has been further extended to deal with some forms of *side-channel* attacks regarding the length of messages [63], and the computation time [64].

5.1.2. Resolution-based approaches

The procedure described in [65] deals with rich user-defined term algebras provided that they can be defined using a convergent rewriting system enjoying the *finite variant property* [66]. This property basically requires that it is possible to finitely pre-compute possible normal forms of terms with variables. This especially includes all subterm convergent equational theo-

ries. In the setting of [65], protocols are modelled as sets of symbolic traces with equality tests. Further, the authors of [65] use first-order Horn clauses to model all possible instantiations of symbolic traces, and they rely on a saturation procedure to put all clauses into *solved forms*. Finally, this finite description of all possible concrete executions is used to decide equivalence between the two processes under study. This procedure is actually able to check an over-approximation (called $\approx_{ct}$) and an under approximation (called $\approx_{ft}$) of trace equivalence, and it has been shown that $\approx_{ct}$ actually coincides with trace equivalence for a large class of processes (the class of determinate processes) that typically includes simple processes. This procedure has been implemented in the tool AKISS[3] and has been effectively tested on several examples including checking vote privacy of an electronic voting protocol relying on the blind signature primitive. Recently, termination of the procedure has been established for subterm convergent theories [67].

Systems we are interested in are highly concurrent and existing methods and tools naively explore all possible symbolic interleavings causing the so called state-explosion problem. This problem seriously limits the practical impact of those tools. Recent works [68, 69] have partially addressed this issue by developing dedicated partial order reduction techniques to dramatically reduce the number of interleavings to explore. They have been implemented in APTE and brought significant speed-up. Actually, they are generic enough to be applicable to any method as long as it performs forward symbolic executions.

5.2. Unbounded number of sessions

The decidability results mentioned in the previous section analyse equivalence for a bounded number of sessions only, that is assuming that protocols are executed a limited number of times. This is of course a strong limitation. Even if no flaw is found when a protocol is executed n times, there is absolutely no guarantee that the protocol remains secure when it is executed $n + 1$ times. Therefore, despite the difficulty of the problem in the general case, several solutions have been proposed for an unbounded number of sessions. A table summarising the main features of existing tools dealing with an unbounded number of sessions is given in Table 2.

[3] http://akiss.gforge.inria.fr

5.2.1. Decidability results

It is well-known that replication (allowing us to encode an unbounded number of sessions) very quickly leads to undecidability even when considering the simple and well-known weak secrecy property. Therefore, obtaining decidability results can only be achieved under various restrictions.

One of the first decidability results for checking trace equivalence of protocols for an unbounded number of sessions is due to Chrétien *et al.* [70, 51]. They consider a limited fragment of protocols, namely *ping-pong protocols*, with standard primitives but pairs, and at most one variable per protocol rules. Even if the secrecy preservation problem is known to be decidable in this setting [71], it turns out that checking equivalence is undecidable. Then, considering determinate protocols, they establish decidability through a characterisation of equivalence of protocols in terms of equality of languages of (generalised, real-time) deterministic pushdown automata. Note that this result only holds for a restricted signature, and names can only be used to produce randomised cipher-texts. Very recently, the algorithm for checking equivalence of deterministic pushdown automata has been implemented [72], and the translation from protocols to pushdown automata has been implemented too, yielding the first prototype able to decide trace equivalence considering an unbounded number of sessions [51].

Assuming finitely many nonces and keys, another decidability result has been obtained in [73]. The primitives that are considered are pairs, and symmetric encryption only, but they go beyond ping-pong protocols and consider the class of simple protocols. In order to derive a strong typing result that drastically limits the shape and size of messages needed to mount an attack, the authors introduce the notion of *type-compliance* for a protocol. This notion generalises the idea of tagging as introduced by Blanchet in [74], and avoids ambiguity in the interpretation of the origin of any message sent on the network. From this typing result, they derive a decision procedure for trace equivalence for an unbounded number of sessions for type-compliant protocols without nonces.

Actually, the typing result mentioned above has also been used to derive the first decidability result for trace equivalence in presence of unlimited fresh nonces [75]. Such a decidability result inherits the conditions introduced above (type-compliance, restricted signature), and develops in addition a notion of *dependency graph*. This notion formally abstracts the dependencies between the actions occurring in a protocol specification. Then, considering

acyclic protocols (*i.e.,* those for which the dependency graph is acyclic) which is intuitively related to protocols without loops in their well-typed executions, decidability of trace equivalence is established. These procedures have not been implemented yet.

5.2.2. Procedures for checking diff-equivalence

As said before, the problem of checking trace equivalence for rich class of protocols is undecidable. To circumvent this undecidability result, many works aim at developing procedures (not necessarily completely automatic) that are sound w.r.t. trace equivalence but not complete. Moreover, termination is not guaranteed. The main idea is to merge the protocols under study into a so-called bi-process, and to consider a strong form of equivalence, namely *diff-equivalence* as described in Definition 6. This method has first been presented in [76] and implemented in the ProVerif tool. Recently, this technique has been integrated into the verification tools Tamarin [77] and Maude-NPA [78] that have been extended to deal with equivalence properties. The main limitation of all these results is the fact that the tools are not able to analyse trace equivalence (but only diff-equivalence). Thus, these tools are not well-suited in general to analyse several privacy-related properties such as (strong) unlinkability [12], and vote privacy [21].

The method presented in [79] and implemented in ProVerif[4] represents bi-processes that are given in input using Horn clauses (performing some well-chosen approximations, and thus losing completeness). Then, a dedicated resolution algorithm tries to resolve those Horn clauses. Cryptographic primitives are decomposed into: reduction rules and a union of linear equational theories (*i.e.,* each equation has the same variables on both sides) and convergent theories (*i.e.,* terminating and confluent). This formalism is flexible enough to model for instance different flavours of encryptions (symmetric, asymmetric, randomised, ...), signature, and blind signature, but excludes exclusive-or, and more generally associative and commutative operators. The resulting tool is quite efficient, and terminates on many examples.

As already mentioned, diff-equivalence is strictly stronger than trace equivalence. Basically, the two processes have to be executed exactly in the same way, notably for internal rules, whereas the attacker cannot observe such details. This problem has been partially tackled in [80] by pushing away

[4]http://prosecco.gforge.inria.fr/personal/bblanche/proverif/

	ProVerif [79]	Maude-NPA [78]	Tamarin [84]
Equivalence	diff-equivalence		
Primitives	linear + convergent	convergent with finite variant (inc. XOR, A.G.)	convergent with finite variant (inc. DH)
Class of protocols	full	linear role with filtering	full + state
Input syntax	applied-pi calculus	strand spaces	multiset rewriting
Termination	may diverge		
Exploration	resolution	backward	

Table 2: Main features of existing tools (for an unbounded number of sessions)

the evaluation of conditionals into terms. Nevertheless, the problem remains in general (*e.g.,* for interleavings of conditionals and observable actions).

To extend the class of equivalences and protocols that can be automatically verified by ProVerif, several extensions have been proposed. For instance, ProSwapper[5]) has been designed to consider cryptographic protocols that require barrier synchronisation (also called phases) to achieve security objectives [81]. The ProSwapper extension allows the algorithm to go beyond diff-equivalence by rearranging bi-processes. This extension has been shown particularly useful to establish vote privacy for several electronic-voting protocols. More recently, theoretical foundations have been provided for this technique and soundness of this extension has been proved [82]. A reduction result to get rid of some particular equations (that cannot be handled by the ProVerif tool) has been devised in [83]. Relying on it, a first automated proof of privacy for the protocol Prêt à Voter (that uses re-encryption and associative/commutative operators) has been carried out with success.

Recently, the approach behind the Tamarin verification tool [85] has been extended to deal with equivalence-based properties. In this approach,

[5]http://www.bensmyth.com/proswapper.php

protocols are modelled as multiset rewriting (MSR) systems. This allows one to model a rich class of protocols that may feature else branches and allow the storage of some data from one session to another. The framework supports a rich term algebra including subterm convergent theories, and Diffie-Hellman exponentiation. The proposed algorithm exploits the finite variant property [66] to get rid of some equations, and it builds on ideas from strands spaces and proof normal forms. It basically performs a backward search from attacks states. TAMARIN provides two ways of constructing proofs: an efficient, fully automated mode that uses heuristics to guide proof search, and an interactive mode. The interactive mode enables the user to explore the proof states using a graphical interface. The TAMARIN tool has been used to analyse different security properties on many protocols. However, regarding equivalence, the tool is less mature and has only been used on a few examples; the main one being a stateful TPM protocol (namely the TPM envelope protocol) on which a strong secrecy property has been established.

The MAUDE-NPA tool has also been recently extended [78] to deal with bi-processes (called *synchronous product*) and diff-equivalence. Their semi-decision procedure is able to deal with a very large class of term algebras (as soon as they have the *finite variant property* as defined in [86]) like Abelian groups, exclusive-or, and exponentiation. However, it can only be applied to linear role scripts with filtering over inputs (and therefore does not handle protocols with else branches). Regarding equivalence, only a few case studies have been performed. The approach of [78] suffers from termination problems, especially when considering primitives such as exclusive-or. Regarding equivalence, their main example is a proof of absence of guessing attacks on a version of the EKE protocol (that relies on standard primitives only).

5.2.3. Some other results

In many cases, existing methods and tools are not sufficient to carry out fully automated proofs. On the other side, fully manual proofs are tedious, error-prone, and hardly verifiable. For instance, previous works gave a manual and formal proof of vote privacy for Helios [87] and the Norwegian e-voting protocol [88]. Those proofs are not automated mainly because existing tools are not able to deal with an unbounded number of voters and complex equational theories featuring for instance homomorphic encryption. For such proofs, one has to exhibit complex bisimulation relations and show static equivalence of infinite families of frames.

Actually, it is also possible to combine manual and automatic proofs. For instance, in [89], the authors establish an unlinkability property on a fixed version of the TMSI reallocation procedure used in mobile telephony systems. As pointed out in the paper, no tool could, at this time, deal with both stateful protocols and an equivalence-based property like unlinkability. Thus, they exhibit a manually-built bisimulation and discharge static equivalence verification to ProVerif.

This idea has also been applied to analyse some privacy properties (namely unlinkability and forward privacy) on a very restricted class of stateful RFID protocols [28]. In this work, single-step protocols that only use hash functions as cryptographic primitives are considered. In such a restricted setting, the authors introduce the notion of frame independence which is closely related to the notion of static equivalence between frames. Then, they provide conditions under which unlinkability and forward privacy hold. They perform several case studies and establish those conditions (up to some arbitrary bound) using ProVerif. More recently, a new method [90] based on sufficient conditions for unlinkability and anonymity allows to automatically verify such security properties for unbounded number of sessions for protocols that were out of the scope of existing tools (*e.g.,* BAC protocol and some RFID protocols). Instead of improving the tools and their precisions, such approaches rather focus on security properties of interest and devise sufficient conditions that are checkable much more easily.

6. Conclusion

The results obtained so far are not completely satisfactory. The protocols that are deployed nowadays in various applications rely on operators and primitives that can still not be handled by existing verification algorithms and tools. For instance, electronic voting protocols often rely on complex primitives in order to achieve their goals (*e.g.* homomorphic encryption [87, 88]), and RFID protocols that have power-consumption constraints often use some low-level operators with algebraic properties (*e.g.* exclusive or operator [91]). Despite some advances that have been done in this direction (see *e.g.* [92]), analysing these protocols is still out of scope of the existing algorithms and tools.

Due to the complexity of the verification problem (especially when considering equivalence-based properties), a promising approach seems to devise methods (with tool support) that are sound but not necessarily complete.

However, we advocate verification techniques that go beyond the notion of diff-equivalence. Learning from previous experience, we deem it acceptable to provide tools with some hints, in order to guide them in their attempt to establish the equivalence property under study. The main difficulty is probably to find a reasonable way to allow interactions between the user and the tool during the verification process.

Acknowledgements. This work has been partially supported by the ANR project Sequoia ANR-14-CE28-0030-01.

7. References

[1] D. Dolev, A. C. Yao, On the security of public key protocols, in: Proc. 22nd Symposium on Foundations of Computer Science (FCS'81), IEEE Computer Society Press, 1981, pp. 350–357.

[2] B. Blanchet, A computationally sound mechanized prover for security protocols, in: Proc. Symposium on Security and Privacy (S&P'06), IEEE Computer Society Press, 2006, pp. 140–154.

[3] M. Abadi, P. Rogaway, Reconciling two views of cryptography (the computational soundness of formal encryption), in: Proc. International Conference on Theoretical Computer Science, 2000, pp. 3–22.

[4] V. Cortier, S. Kremer, B. Warinschi, A survey of symbolic methods in computational analysis of cryptographic systems, Journal of Automated Reasoning 46 (3-4) (2011) 225–259.

[5] G. Bana, H. Comon-Lundh, Towards unconditional soundness: Computationally complete symbolic attacker, in: Proc. 1st International Conference on Principles of Security and Trust (POST'12), Springer, 2012, pp. 189–208.

[6] G. Bana, H. Comon-Lundh, A computationally complete symbolic attacker for equivalence properties, in: Proc. 21st Conference on Computer and Communications Security (CCS'14)), ACM, 2014, pp. 609–620.

[7] V. Cortier, F. Eigner, S. Kremer, M. Maffei, C. Wiedling, Type-based verification of electronic voting protocols, in: Proc. 4th Conference on Principles of Security and Trust (POST'15), Springer, 2015, pp. 303–323.

[8] G. Barthe, C. Fournet, B. Grégoire, P.-Y. Strub, N. Swamy, S. Zanella-Béguelin, Probabilistic relational verification for cryptographic implementations, in: Proc. 41st Symposium on Principles of Programming Languages (POPL'14), Vol. 49, ACM, 2014, pp. 193–206.

[9] W. Diffie, M. Hellman, New directions in cryptography, Transactions on Information Society 22 (6) (1976) 644–654.

[10] R. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM 21 (2) (1978) 120–126.

[11] PKI for machine readable travel documents offering ICC read-only access, Tech. rep., International Civil Aviation Organization (2004).

[12] M. Arapinis, T. Chothia, E. Ritter, M. Ryan, Analysing unlinkability and anonymity using the applied pi calculus, in: Proc. 23rd Computer Security Foundations Symposium (CSF'10), IEEE Computer Society Press, 2010, pp. 107–121.

[13] G. Lowe, A hierarchy of authentication specifications, in: Proc. 10th Computer Security Foundations Workshop (CSFW'97), IEEE Computer Society Press, 1997, pp. 18–30.

[14] M. Abadi, Secrecy by typing in security protocols, in: Theoretical Aspects of Computer Software, Springer, 1997, pp. 611–638.

[15] B. Blanchet, Automatic proof of strong secrecy for security protocols, in: Proc. . 2004 Symposium on Security and Privacy, IEEE Computer Society Press, 2004, pp. 86–100.

[16] M. Abadi, A. D. Gordon, A calculus for cryptographic protocols: The spi calculus, in: Proc. of the 4th ACM conference on Computer and communications security, ACM, 1997, pp. 36–47.

[17] M. Arapinis, T. Chothia, E. Ritter, M. Ryan, Untraceability in the applied pi-calculus, in: Proc. International Conference for Internet Technology and Secured Transactions (ICITST,09), IEEE Computer Society Press, 2009, pp. 1–6.

[18] S. Schneider, A. Sidiropoulos, CSP and anonymity, in: Proc. International Conference on Computer Security (ESORICS'96), Springer, 1996, pp. 198–218.

[19] T. Chothia, Analysing the mute anonymous file-sharing system using the pi-calculus, in: Formal Techniques for Networked and Distributed Systems-FORTE 2006, Springer, 2006, pp. 115–130.

[20] S. Kremer, M. D. Ryan, Analysis of an electronic voting protocol in the applied pi-calculus, in: Proc. 14th European Symposium on Programming (ESOP'05), Vol. 3444 of LNCS, Springer-Verlag, 2005, pp. 186–200.

[21] S. Delaune, S. Kremer, M. D. Ryan, Verifying privacy-type properties of electronic voting protocols, Journal of Computer Security 17 (4) (2008) 435–487.

[22] J. Benaloh, D. Tuinstra, Receipt-free secret-ballot elections (extended abstract), in: Proc. 26th Symposium on Theory of Computing (STOC'94), ACM Press, 1994, pp. 544–553.

[23] S. Delaune, S. Kremer, M. D. Ryan, Coercion-resistance and receipt-freeness in electronic voting, in: Proc. 19th Computer Security Foundations Workshop (CSFW'06), IEEE Computer Society Press, 2006, pp. 28–39.

[24] M. Backes, C. Hritcu, M. Maffei, Automated verification of remote electronic voting protocols in the applied pi-calculus, in: Proc. 21st Computer Security Foundations Symposium, (CSF'08), IEEE Computer Society Press, 2008, pp. 195–209.

[25] M. Arapinis, V. Cortier, S. Kremer, M. D. Ryan, Practical Everlasting Privacy, in: Proc. 2nd Conference on Principles of Security and Trust (POST'13), Vol. 7796 of LNCS, Springer, 2013, pp. 21–40.

[26] N. Dong, H. Jonker, J. Pang, Analysis of a receipt-free auction protocol in the applied pi calculus, in: Proceedings of the 7th International Workshop on Formal Aspects in Security and Trust (FAST'10), Vol. 6561 of LNCS, Springer, 2010, pp. 223–238.

[27] J. Dreier, P. Lafourcade, Y. Lakhnech, Formal verification of e-auction protocols, in: Proc. 2nd Conferences on Principles of Security and Trust (POST'13), Vol. 7796 of LNCS, Springer, 2013, pp. 247–266.

[28] M. Brusó, K. Chatzikokolakis, J. Den Hartog, Formal verification of privacy for RFID systems, in: Proc. 23rd Computer Security Foundations Symposium (CSF'10), IEEE Computer Society Press, 2010, pp. 75–88.

[29] T. Van Deursen, S. Mauw, S. Radomirović, Untraceability of RFID protocols, in: Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks, Springer, 2008, pp. 1–15.

[30] M. Brusó, K. Chatzikokolakis, S. Etalle, J. Den Hartog, Linking unlinkability, in: Trustworthy Global Computing, Springer, 2013, pp. 129–144.

[31] R. Canetti, Universally composable security: A new paradigm for cryptographic protocols, in: Proc. 42nd Annual Symposium on Foundations of Computer Science (FOCS'01), IEEE Computer Society Press, 2001, pp. 136–145.

[32] S. Delaune, S. Kremer, O. Pereira, Simulation based security in the applied pi calculus, in: Proc. 29th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'09), Vol. 4 of Leibniz International Proc. in Informatics, Leibniz-Zentrum für Informatik, 2009, pp. 169–180.

[33] F. Böhl, D. Unruh, Symbolic universal composability, in: Proc. 26th Computer Security Foundations Symposium (CSF'13), IEEE Computer Society Press, 2013, pp. 257–271.

[34] M. Abadi, C. Fournet, Mobile values, new names, and secure communication, in: Proc. 28th Symposium on Principles of Programming Languages (POPL'01), ACM Press, 2001, pp. 104–115.

[35] B. Blanchet, Proverif 1.91, http://prosecco.gforge.inria.fr/personal/bblanche/, as downloaded on October 1st, 2015. See files in directory /examples/pitype/choice/.

[36] M. Abadi, 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, Springer-Verlag, 2004, pp. 46–58.

[37] M. Abadi, V. Cortier, Deciding knowledge in security protocols under (many more) equational theories, in: Proc. 18th Computer Security Foundations Workshop (CSFW'05), IEEE Computer Society Press, 2005, pp. 62–76.

[38] V. Cortier, S. Delaune, Decidability and combination results for two notions of knowledge in security protocols, Journal of Automated Reasoning 48 (4) (2012) 441–487.

[39] Ş. Ciobâcă, S. Delaune, S. Kremer, Computing knowledge in security protocols under convergent equational theories, Journal of Automated Reasoning 48 (2) (2012) 219–262.

[40] M. Baudet, V. Cortier, S. Delaune, YAPA: A generic tool for computing intruder knowledge, ACM Transactions on Computational Logic 14 (1:4).

[41] B. Conchinha, D. A. Basin, C. Caleiro, FAST: an efficient decision procedure for deduction and static equivalence, in: Proc. 22nd International Conference on Rewriting Techniques and Applications, (RTA'11), Vol. 10 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 11–20.

[42] M. Abadi, A. D. Gordon, A calculus for cryptographic protocols: The spi calculus, Information and Computation 148 (1) (1999) 1–70.

[43] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, A. Scedrov, A metanotation for protocol analysis, in: Proc. 12th Computer Security Foundations Workshop (CSFW'99), IEEE Computer Society Press, 1999, pp. 55–69.

[44] F. J. Thayer, J. C. Herzog, J. D. Guttman, Strand spaces: Proving security protocols correct, Journal of Computer Security 7 (1) (1999) 191–230.

[45] R. Milner, D. Sangiorgi, Barbed bisimulation, in: W. Kuich (Ed.), Proc. 19th International Colloquium on Automata, Languages, and Programming (ICALP'92), Vol. 623 of LNCS, Springer Verlag, 1992, pp. 685–695.

[46] M. Abadi, A. D. Gordon, A bisimulation method for cryptographic protocols, Nord. J. Comput. 5 (4) (1998) 267.

[47] M.Boreale, R. D. Nicola, R. Pugliese, Proof techniques for cryptographic processes, SIAM Journal on Computing 31 (3) (2002) 947–986.

[48] J. Bengtson, M. Johansson, J. Parrow, B. Victor, Psi-calculi: a framework for mobile processes with nominal data and logic, Logical Methods in Computer Science 7 (1).

[49] V. Cheval, V. Cortier, S. Delaune, Deciding equivalence-based properties using constraint solving, Theoretical Computer Science 492 (2013) 1–39.

[50] H. Hüttel, Deciding framed bisimilarity, Electronic Notes in Theoretical Computer Science 68 (6) (2003) 1–18.

[51] R. Chrétien, V. Cortier, S. Delaune, From security protocols to pushdown automata, ACM Transactions on Computational Logic 17 (1:3).

[52] S. Delaune, S. Kremer, M. D. Ryan, Symbolic bisimulation for the applied pi calculus, Journal of Computer Security 18 (2) (2010) 317–377.

[53] L. Durante, R. Sisto, A. Valenzano, Automatic testing equivalence verification of spi calculus specifications, ACM Transactions on Software Engineering and Methodology 12 (2) (2003) 222–284.

[54] J. Borgström, A complete symbolic bisimilarity for an extended spi calculus, Electr. Notes Theor. Comput. Sci. 242 (3) (2009) 3–20.

[55] J. Borgström, R. Gutkovas, I. Rodhe, B. Victor, The psi-calculi workbench: A generic tool for applied process calculi, ACM Trans. Embedded Comput. Syst. 14 (1) (2015) 9:1–9:25.

[56] M. Baudet, Deciding security of protocols against off-line guessing attacks, in: Proc. 12th ACM conference on Computer and communications security (CCS'05), ACM, 2005, pp. 16–25.

[57] Y. Chevalier, M. Rusinowitch, Decidability of symbolic equivalence of derivations, Journal of Automated Reasoning 48 (2) (2012) 263–292.

[58] A. Tiu, J. Dawson, Automating open bisimulation checking for the spi calculus, in: Proc. 23rd Computer Security Foundations Symposium (CSF'10), IEEE Computer Society Press, 2010, pp. 307–321.

[59] A. Tiu, A trace based bisimulation for the spi calculus, in: Programming Languages and Systems, Springer, 2007, pp. 367–382.

[60] V. Cheval, H. Comon-Lundh, S. Delaune, Trace equivalence decision: Negative tests and non-determinism, in: Proc. 18th ACM Conference on Computer and Communications Security (CCS'11), ACM Press, 2011, pp. 321–330.

[61] V. Cheval, H. Comon-Lundh, S. Delaune, Automating security analysis: symbolic equivalence of constraint systems, in: Proc. 5th International Joint Conference on Automated Reasoning (IJCAR'10), Vol. 6173 of LNAI, Springer-Verlag, 2010, pp. 412–426.

[62] V. Cheval, Apte: an algorithm for proving trace equivalence, in: Proc. 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14), Vol. 8413 of LNCS, 2014, pp. 587–592.

[63] V. Cheval, V. Cortier, A. Plet, Lengths may break privacy – or how to check for equivalences with length, in: Proc. 25th International Conference on Computer Aided Verification (CAV'13), Vol. 8044 of LNCS, Springer Berlin Heidelberg, 2013, pp. 708–723.

[64] V. Cheval, V. Cortier, Timing attacks in security protocols: symbolic framework and proof techniques, in: Proc. 4th Conference on Principles of Security and Trust (POST'15), Springer, 2015, pp. 280–299.

[65] R. Chadha, Ş. Ciobâcă, S. Kremer, Automated verification of equivalence properties of cryptographic protocols, in: Proc. European Symposium on Programming (ESOP'12), Springer, 2012, pp. 108–127.

[66] H. Comon-Lundh, S. Delaune, The finite variant property: How to get rid of some algebraic properties, in: Proc. International Conference on Rewriting Techniques and Applications (RTA'05), Springer, 2005, pp. 294–307.

[67] R. Chadha, V. Cheval, Ş. Ciobâcă, S. Kremer, Automated verification of equivalence properties of cryptographic protocol, ACM Transactions on Computational LogicTo appear.
URL `https://hal.inria.fr/hal-01306561/document`

[68] D. Baelde, S. Delaune, L. Hirschi, Partial order reduction for security protocols, in: Proc. 26th International Conference on Concurrency Theory (CONCUR'15), Vol. 42 of LIPIcs, Leibniz-Zentrum für Informatik, 2015, pp. 497–510.

[69] D. Baelde, S. Delaune, L. Hirschi, A reduced semantics for deciding trace equivalence using constraint systems, in: Proc. 3rd Conference on Principles of Security and Trust (POST'14), Springer, 2014, pp. 1–21.

[70] R. Chrétien, V. Cortier, S. Delaune, From security protocols to pushdown automata, in: Proc. 40th International Colloquium on Automata, Languages and Programming (ICALP'13), Vol. 7966 of LNCS, Springer, 2013, pp. 137–149.

[71] H. Comon-Lundh, V. Cortier, New decidability results for fragments of first-order logic and application to cryptographic protocols, in: 14th Proc. International Conference on Rewriting Techniques and Applications (RTA'2003), Vol. 2706 of LNCS, Springer, 2003, pp. 148–164.

[72] P. Henry, G. Sénizergues, Lalblc a program testing the equivalence of dpda's, in: Proc. 18th International Conference on Implementation and Application of Automata (CIAA'13), Vol. 7982 of LNCS, Springer, Halifax, NS, Canada, 2013, pp. 169–180.

[73] R. Chrétien, V. Cortier, S. Delaune, Typing messages for free in security protocols: the case of equivalence properties, in: Proc. 25th International Conference on Concurrency Theory (CONCUR'14), Vol. 8704 of LNCS, Springer, 2014, pp. 372–386.

[74] B. Blanchet, A. Podelski, Verification of cryptographic protocols: Tagging enforces termination, in: Proc. International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'03), Vol. 2620 of LNCS, 2003, pp. 136–152.

[75] R. Chrétien, V. Cortier, S. Delaune, Decidability of trace equivalence for protocols with nonces, in: Proc. 28th Computer Security Foundations Symposium (CSF'15), IEEE Computer Society Press, 2015, pp. 170–184.

[76] B. Blanchet, M. Abadi, C. Fournet, Automated Verification of Selected Equivalences for Security Protocols, in: Proc. 20th Symposium on Logic

in Computer Science (LICS'05), IEEE Computer Society Press, 2005, pp. 331–340.

[77] D. Basin, J. Dreier, R. Sasse, Automated symbolic proofs of observational equivalence, in: Proc. 22nd Conference on Computer and Communications Security (CCS'15), ACM, 2015, pp. 1144–1155.

[78] S. Santiago, S. Escobar, C. Meadows, J. Meseguer, A formal definition of protocol indistinguishability and its verification using Maude-NPA, in: Security and Trust Management, Springer, 2014, pp. 162–177.

[79] B. Blanchet, M. Abadi, C. Fournet, Automated verification of selected equivalences for security protocols, Journal of Logic and Algebraic Programming 75 (1) (2008) 3–51.

[80] V. Cheval, B. Blanchet, Proving more observational equivalences with ProVerif, in: Proc. 2nd Conference on Principles of Security and Trust (POST'13), Vol. 7796 of LNCS, Springer, 2013, pp. 226–246.

[81] S. Delaune, M. D. Ryan, B. Smyth, Automatic verification of privacy properties in the applied pi-calculus, in: Proc. 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security (IFIPTM'08), Vol. 263 of IFIP Conference Proc. Springer, 2008, pp. 263–278.

[82] B. Blanchet, B. Smyth, Automated reasoning for equivalences in the applied pi calculus with barriers, in: Proc. 29th Computer Security Foundations Symposium (CSF'16), 2016, to appear.

[83] M. Arapinis, S. Bursuc, M. D. Ryan, Reduction of equational theories for verification of trace equivalence: Re-encryption, associativity and commutativity, in: Proc. 1st International Conference on Principles of Security and Trust (POST'12), Vol. 7215 of LNCS, Springer, 2012, pp. 169–188.

[84] S. Meier, B. Schmidt, C. Cremers, D. Basin, The tamarin prover for the symbolic analysis of security protocols, in: Proc. International Conference on Computer Aided Verification (CAV'13), Springer, 2013, pp. 696–701.

[85] B. Schmidt, S. Meier, C. Cremers, D. Basin, Automated analysis of Diffie-Hellman protocols and advanced security properties, in: Proc.

25th Computer Security Foundations Symposium (CSF'12), IEEE Computer Society Press, 2012, pp. 78–94.

[86] S. Escobar, R. Sasse, J. Meseguer, Folding variant narrowing and optimal variant termination, in: Rewriting Logic and Its Applications, Springer, 2010, pp. 52–68.

[87] V. Cortier, B. Smyth, Attacking and fixing helios: An analysis of ballot secrecy, in: Proc. 24th Computer Security Foundations Symposium (CSF'11), IEEE Computer Society Press, 2011, pp. 297–311.

[88] V. Cortier, C. Wiedling, A formal analysis of the norwegian e-voting protocol, in: Proc. 2nd Conference on Principles of Security and Trust, Springer, 2012, pp. 109–128.

[89] M. Arapinis, L. I. Mancini, E. Ritter, M. Ryan, Privacy through pseudonymity in mobile telephony systems, in: Proc. Network and Distributed System Security Symposium (NDSS'14), 2014.

[90] L. Hirschi, D. Baelde, S. Delaune, A method for verifying privacy-type properties: the unbounded case, in: Proc. 37th Symposium on Security and Privacy (S&P'16), 2016, to appear.

[91] T. van Deursen, S. Radomirović, Algebraic attacks on RFID protocols, in: Information Security Theory and Practice. Smart Devices, Pervasive Systems, and Ubiquitous Networks, Springer, 2009, pp. 38–51.

[92] S. Delaune, S. Kremer, D. Pasailă, Security protocols, constraint systems, and group theories, in: Proc. 6th International Joint Conference on Automated Reasoning (IJCAR'12), Vol. 7364 of LNAI, Springer-Verlag, 2012, pp. 164–178.