Analysing privacy-type properties in cryptographic protocols*

Stéphanie Delaune
Univ Rennes, CNRS, IRISA, France
stephanie.delaune@irisa.fr

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. For example, passports are no more pure paper documents. Instead, they contain a chip that stores additional information such as pictures and fingerprints of their holder. In order to ensure privacy, these chips include a mechanism, i.e. a cryptographic protocol, that does not let the passport disclose private information to external users except the intended terminal. This is just a single example but of course privacy appears in many other contexts such as RFIDs technologies or electronic voting.

Formal methods have been successfully applied to automatically analyse traditional protocols and discover their flaws. Privacy-type security properties (e.g. anonymity, unlinkability, vote secrecy, ...) are expressed relying on a notion of behavioural equivalence, and are actually more difficult to analyse than confidentiality and authentication properties. We will discuss some recent advances that have been done to analyse automatically equivalence-based security properties, and we will review some issues that remain to be solved in this field.

2012 ACM Subject Classification Security and privacy – Formal methods and theory of security – Logic and verification

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

Digital Object Identifier 10.4230/LIPIcs.FSCD.2018.1

1 Introduction

Cryptographic protocols are widely used today to secure communications with the aim of achieving various security goals. For instance, TLS (Transport Layer Security) is a protocol that is widely used to provide authentication and encryption in order to send sensitive data such as credit card numbers to a vendor. Those protocols use cryptographic primitives as building blocks such as encryptions, signatures, and hashes.

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 due to some *logical attacks* coming from the poor design of the protocol itself. As an example, we can cite the well-known *man-in-the-middle attack* on the Needham Schroeder protocol [55] that has been discovered by Lowe seventeen years after the publication of the original protocol [53]. This is just a single example for which Lowe proposed a simple fix: the second message $\{N_a, N_b\}_{\mathsf{pub}(B)}$ is replaced by $\{B, N_a, N_b\}_{\mathsf{pub}(B)}$ in the fixed version

* This work has been partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 714955-POPSTAR) and the ANR project TECAP.

of the protocol, i.e. the name of the sender has been simply added into the ciphertext. Such a modification was sufficient to discard the man-in-the-middle attack and to prove the protocol secure. Even if protocols are relatively small programs, they are rather difficult to analyse and the difference between a secure protocol and an insecure one may be rather subtle. For instance, replacing the second message by $\{N_a, N_b, B\}_{\mathsf{pub}(B)}$, i.e. the name of B is now appended at the end of the original message, results in a protocol on which a man-in-the-middle attack similar to the one discovered by Lowe is again possible (see [37] for a description of this attack).

One successful approach when designing and analysing security protocols, is the use of formal methods. The purpose of formal verification is to provide rigorous frameworks and techniques to analyse protocols and find their flaws. For example, a flaw has been discovered in the Single-Sign-On protocol used e.g. by Google Apps. It has been shown that a malicious application could very easily get access to any other application (e.g. Gmail or Google Calendar) of their users [6]. This flaw has been found when analysing the protocol using the Avantssar validation platform [7]. Another example is a flaw on vote secrecy discovered during the formal and manual analysis of an electronic voting protocol [42]. All these results have been obtained using formal symbolic models, where most of the cryptographic details are ignored using abstract structures, and the communication network is assumed to be entirely controlled by an omniscient attacker. Although less precise than computational models used by cryptographers, this symbolic approach benefits from automation and can thus target more complex protocols and scenarios than those analysed using the computational approach. The techniques used in symbolic models have become mature and several tools for protocol verification are nowadays available, e.g. Avantssar platform [7], ProVerif [15], and Tamarin [58].

Most of the results existing in this field focus on reachability properties such as authentication or secrecy: for any execution of the protocol, it should never be the case that an attacker learns some secret (confidentiality property) or that an attacker makes Alice think she's talking to Bob while Bob did not engage a conversation with her (authentication property). However, privacy properties such as vote secrecy, anonymity, or untraceability cannot be expressed as reachability properties. Formally the behaviour of a protocol can be modelled through a process algebra such as the pi-calculus, enriched with terms to model cryptographic messages. Then, privacy-type properties are expressed relying on a notion of behavioural equivalence between processes. For example, Alice's identity remains private if an attacker cannot distinguish a session where Alice is talking from a session where Bob is talking. As mentioned above, many results and tools have been developed in the context of reachability properties. Results for equivalence properties are more rare but a lot of attention has been devoted to its study during the ten past years.

In this paper, we will review existing results and tools dedicated to the study of equivalence-based properties. We will present some recent advances that have been done in this area, and discuss some challenges that remain to be solved.

2 Some examples

We briefly describe in this section some cryptographic protocols on which privacy-type properties are particularly relevant. For illustrative purposes, we first consider a rather simple RFID protocol following a description given in [61] before explaining two protocols coming from the e-passport application: the BAC protocol and its successor the PACE protocol.

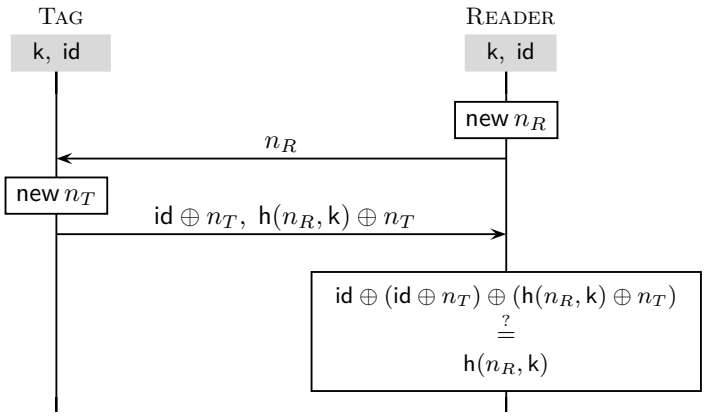

Figure 1 Description of an RFID protocol due to Kim *et al.* [51].

2.1 A simple RFID protocol

To illustrate our formalism along this paper, we will consider a rather simple RFID protocol proposed by Kim et al. [51] in 2007. We follow the description given in [61]. In this protocol, the reader and the tag id share a secret symmetric key k. The protocol does not rely on any encryption algorithm but instead uses a hash function, denoted h, and the exclusive or operator, denoted $\oplus$, which is commutative and associative. Moreover, it has the property that equal terms cancel each other out, i.e. $t \oplus t = 0$ where 0 is the neutral element.

The reader starts by sending a nonce, i.e. a fresh random number n_R. Once it receives this first message, the tag generates its own nonce n_T and computes its answer relying on the hash function and the exclusive-or operator. When receiving this second message, the reader will be able to retrieve n_T from the first component by cancelling out the value id. Then, it will xor this value with the second component and check whether the result is equal to $\mathsf{h}(n_R, \mathsf{k})$. Note that since the reader knows n_R and k, it can indeed easily compute the message $\mathsf{h}(n_R, \mathsf{k})$.

The aim of this protocol is not only to authenticate the tag but also to ensure its unlinkability. An attacker should not be able to observe whether he has seen the same tag twice or two different tags. Actually, this unlinkability property is not satisfied. An attacker can simply send his own nonce n_R^0 and infer whether the tag in presence is the same or not from the message he received. For this, the attacker simply apply the exclusive-or operator on the two components of the message sent by the tag. The result of this operation is $\mathsf{id} \oplus \mathsf{h}(n_R^0, \mathsf{k})$ and once n_R^0 is fixed, this value only depends on the identity of the tag.

We will formalise this later on as an indistinguishability property, relying on the notion of trace equivalence.

2.2 Electronic passport

An e-passport is no more a pure paper document but instead contains an RFID chip that stores the critical information printed on the passport. The International Civil Aviation Organisation (ICAO) standard specifies several protocols through which this information can be accessed. In particular, access to the data stored on the passport are protected by

the Basic Access Control (BAC) protocol, or now its successor the Password Authenticated Connection Establishment (PACE) protocol.

BAC protocol

This is a password-based authenticated key exchange protocol (see Figure 2) whose security relies on two master keys, namely ke and km. Actually, before executing the BAC protocol, the reader optically scans a low entropy secret from which these two keys ke and km are derived. Thus, these keys are symmetric keys shared between the passport (the RFID tag) and the reader. Then, the BAC protocol establishes a key seed from which two sessions keys *kenc* and *kmac* are derived. The session keys are then used to prevent skimming and eavesdropping on the subsequent communication with the e-passport. In particular, they are used to encrypt and mac the messages exchanged during the execution of the subsequent protocols.

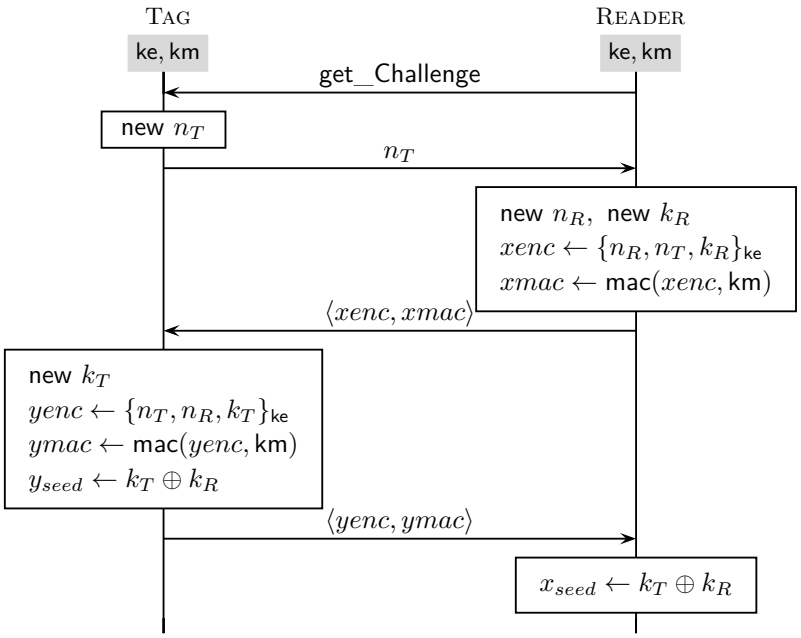

Figure 2 Basic Access Control protocol

First, we may note that the nonces n_R and n_T are not placed in the same order in the two ciphertexts: $\{n_R, n_T, k_R\}_{\mathsf{ke}}$ and $\{n_T, n_R, k_T\}_{\mathsf{ke}}$. Actually, this is not an insignificant choice. This choice prevents a replay attack which would be possible otherwise since it would be possible for an attacker to answer to the message send by a reader without knowing the keys ke and km. Indeed, an attacker could simply replay the message he just received, and this will lead to the computation of the seed $x_{seed} = k_R \oplus k_R = 0$.

Second, the low entropy secret printed in the first page of a passport and from which the keys ke and km are derived makes the BAC protocol vulnerable to off-line guessing attacks. Indeed, an attacker who listens to the communication will learn e.g. $\{n_R, n_T, k_R\}_{\mathsf{ke}}$. Then, he can simply try to decrypt this ciphertext using all possible values for ke until he finds a

value that allows him to obtain n_T (nonce that has been sent in clear and that is therefore known by the attacker).

Third, we may note that when the passport receives an incorrect message $\langle xenc, xmac\rangle$, the behaviour of the passport is not specified, Due to this, some implementations of the BAC protocol breaches unlinkability [29]: in the french implementation, the passport tag replies different error messages depending on whether the problem comes from an incorrect mac or an incorrect nonce (i.e. the nonce n_T inside the ciphertext is not the one previously generated by the passport). An attacker could then trace a passport (without knowing the keys ke and km) in the following way:

1. he listens to a first session between a reader and a tag T and store $m = \langle xenc, xmac\rangle$;
2. then, in a different session, he sends the message m and wait for the tag's response;
   a. if he receives a nonce error then he knows that the tag succeeded to mac $xenc$ with his own key ke and so this tag is T;
   b. if he receives a mac error then he knows that the tag is not T.

This gives the attacker a way to distinguish between two different passports. Such a flaw does not exist in other implementations where the same error messages is sent in both cases.

PACE protocol

The Password Authenticated Connection Establishment protocol [56] (PACE) has been proposed by the Bundesamt für Sicherheit in der Informationstechnik (BSI) to replace the BAC protocol. Similarly to BAC, the purpose of PACE is to establish a secure channel based on an optically-scanned key k. A description is given in Figure 3. The tag and the reader perform a first Diffie-Hellman exchange and derive G. Then, they perform a second Diffie-Hellman exchange based on the parameter G computed at the previous step, and they derive a session key k'. In a final stage, they confirm the values that have been exchanged using message authentication codes.

First, we may note that the low entropy of the secret k is not a problem anymore assuming that the decryption operation on the ciphertext $\{s_T\}_{\mathsf{k}}$ will not fail when the key used to decrypt is not k. This means that the resulting computation $\mathsf{sdec}(\{s_T\}_{\mathsf{k}}, \mathsf{k}_I)$ will be a valid message even if $\mathsf{k} \neq \mathsf{k}_I$, and thus the protocol will pursue its execution normally.

Second, we would like to comment on the disequality test performed by the reader. Such a test is important to prevent an attacker to execute with success the PACE protocol. Without such a test, an attacker can eavesdrop a message $\{s_T\}_{\mathsf{k}}$ from an honest session, and then reuse it to execute a session with a reader. He simply has to send the ciphertext, and then answer to the reader by replaying the message he just received. This means that the attacker would not have to know k to successfully execute the protocol whereas he is supposed to know it to compute G. Of course, this directly leads to an authentication issue that can be turned into a linkability attack.

Third, the fact that the format of the two last messages are similar is surprising. Due to this, an attacker can send $\{s_T\}_k$ (eavesdrop during a previous session) to two different readers and then simply forward the messages from one reader to another. Both readers will be able to compute the two rounds of Diffie-Hellman, and the mac verification phase will not prevent this behaviour. Even if in practice, this flaw seems hard to exploit, it could be a real privacy concern in some other contexts. Actually, as proposed in [48], this flaw can be fixed by adding tags in the two last messages in order to avoid confusions between reader's messages and tag's messages.

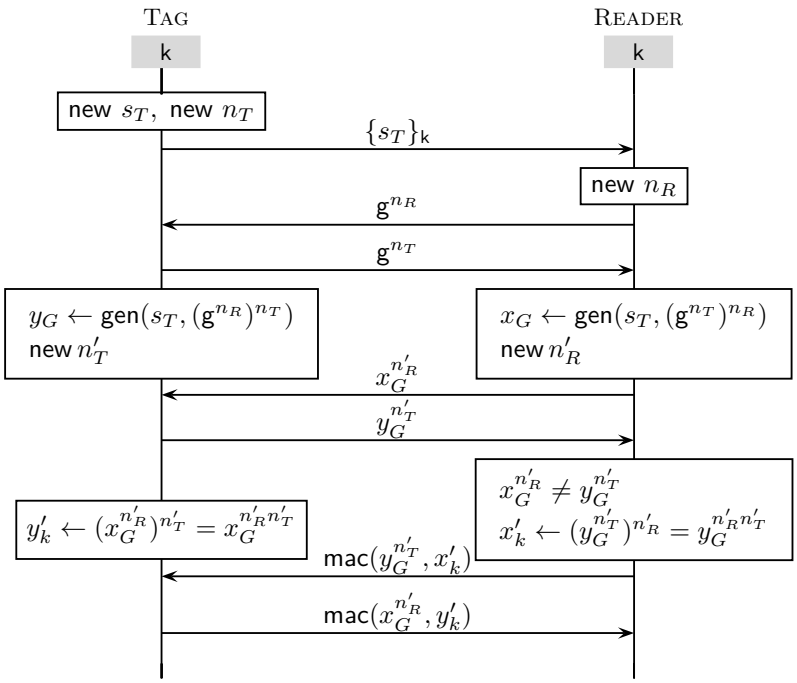

Figure 3 Password Authenticated Connection Establishment protocol

3 Modelling protocols

Several symbolic models have been proposed for cryptographic protocols. The first one has been described by Dolev and Yao [45] and several other models have been proposed since then (e.g. strand spaces [59], multiset rewriting [19], spi-calculus [3]). A unified model would enable better comparisons between the different existing results but unfortunately such a model does not exist currently. Nevertheless, all existing models share some common features: messages are modelled using first-order terms, and they propose some constructions for modelling communication and taking into account the concurrency nature of these programs.

3.1 Messages as terms

In symbolic models, messages are a key concept. Whereas messages are bitstrings in the real-world (and in the computational approach as well), they are modelled using first-order terms within the symbolic model. Formally, we consider an infinite set $\mathcal{N}$ of *names* to represent atomic data such as keys, nonces, and we also consider two infinite sets of *variables* $\mathcal{X}$ and $\mathcal{W}$. The variables in $\mathcal{X}$ are used to model unknown parts of the messages expected by a participant, whereas variables in $\mathcal{W}$, called *handles*, are used as pointers. They refer to messages that have been previously sent on the network and that are therefore known to the attacker.

To model cryptographic primitives, such as encryptions, signatures, hashes, *etc*, we rely on function symbols, namely a *signature*, that allows one to build terms representing mes-

sages sent over the network by the participants. The set of terms built from a set of atomic data A by applying function symbols in a signature Σ are denoted $\mathcal{T}(\Sigma, A)$.

▶ **Example 1.** To model the BAC and the RFID protocols described in Section 2, we may consider the signature:

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

The function symbols senc and sdec (both of arity 2) represent symmetric encryption, whereas $\langle\,\rangle$ (arity 2) is used to concatenate messages. The two components of such an operator can be retrieved using the projection functions proj_1 and proj_2 (both of arity 1). We also consider a function symbol to model an hash function, the symbol h (arity 1), as well as a function symbol mac (arity 2) to model message authentication codes. Lastly, the function symbol $\oplus$ (arity 2) is used to model the exclusive-or operator, and the constant 0 is its neutral element.

Then, we assign a meaning to the function symbols through an equational theory. Formally, we consider a set of equations between terms (without names), and we denote $=_{\mathsf{E}}$ the smallest congruence relation which is closed under the substitution of terms for variables.

▶ **Example 2.** Going back to our previous example, we will typically consider the following set E_{ex} of equations:

$$\begin{array}{rclrclrcl} \mathsf{proj}_1(\langle x, y\rangle) & = & x & x \oplus (y \oplus z) & = & (x \oplus y) \oplus z & x \oplus 0 & = & x \\ \mathsf{proj}_2(\langle x, y\rangle) & = & y & x \oplus y & = & y \oplus x & x \oplus x & = & 0 \\ \mathsf{sdec}(\mathsf{senc}(x, y), y) & = & x & & & & & & \end{array}$$

Considering $m = \mathsf{senc}(\langle n_R, \langle n_T, k_R\rangle\rangle, \mathsf{ke})$, we have that $\mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(m, \mathsf{ke}))) =_{\mathsf{E}_{\mathsf{ex}}} n_T$. We may note that the symbols mac and h are not involved in any equation. Those primitives are modelled using free function symbols since they are one-way functions which are typically assumed to be collision resistant.

Sometimes, function symbols are split into two sets: *constructors* and *destructors*. In such a case, a rewriting system is used to assign a meaning to the function symbols. Constructors symbols, typically senc, $\langle\,\rangle$, etc are used to build messages, whereas destructor symbols, such as sdec, proj_1, and proj_2, are only there to perform computations meaning that a rewriting rule has to apply to make them disappear. If no rewriting rule applies, and the destructor is still there, it means that the computation fails, and the resulting term is not considered as a message. This gives us two slightly different ways to model e.g. symmetric encryption. Both are useful when modelling protocols depending on the properties of the encryption scheme. For instance, going back to the PACE protocol, it is important here to model encryption relying on an equation to take into account the fact that $\mathsf{sdec}(\mathsf{senc}(s_T, \mathsf{k}), \mathsf{k}')$ is a computation that does not lead to a failure but instead gives a result, i.e. a message, and the reader will proceed with the resulting value.

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 (i.e. terms without variable) $u_1, \ldots, u_n$. This means that he knows all messages and also their order. When analysing equivalence-based security properties, it is not enough to say that the attacker knows the set of terms $\{u_1, \ldots, u_n\}$. In the applied-pi calculus [2], such a sequence of messages is organised into a *frame*, i.e. a substitution of the form:

$$\phi = \{w_1 \mapsto u_1, \ldots, w_n \mapsto u_n\}.$$

The handle $w_i \in \mathcal{W}$ enables us to refer to the message u_i, and these variables will allow us to make explicit the order in which these messages are sent. Given a frame ϕ, we denote $dom(\phi)$ its *domain*, i.e. $dom(\phi) = \{w_1, \ldots, w_n\}$.

We need also to consider computations feasible by an attacker. We call such a computation a *recipe*. Formally, a recipe is a term built from (public) function symbols and handles from $\mathcal{W}$.

▶ **Definition 3.** Given a frame ϕ and a term $u \in \mathcal{T}(\Sigma, \mathcal{N})$, we say that u is *deducible* from ϕ, denoted $\phi \vdash_{\mathsf{E}} u$, when there exists a recipe R, i.e. a term in $\mathcal{T}(\Sigma, dom(\phi))$, such that $R\phi =_{\mathsf{E}} u$.

3.2 Protocols as processes

A popular way to model protocols is to rely on a process algebra. Several calculus have been proposed to model protocols, e.g. CSP [49], spi-calculus [3], applied-pi calculus [2]. They have similar constructs as those in the pi-calculus introduced by Milner in 1999 [54]. However, instead of exchanging atomic data, terms that are exchanged are first-order terms. This allows us to model in a more faithful way cryptographic protocols that use cryptographic messages. Typically, considering a set $\mathcal{C}h$ of public channel names, processes are generated by a grammar as follows:

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

where $u_1, u_2, u \in \mathcal{T}(\Sigma, \mathcal{N} \cup \mathcal{X})$, $c \in \mathcal{C}h$, and $n \in \mathcal{N}$.

Most of the constructions are rather standard in process algebra. As usual, the null process, denoted 0, represents the process that does nothing. Such a process is often omitted for sake of conciseness. The process $P \mid Q$ runs P and Q in parallel meaning that we do not know in which order the actions of P and Q will be done. All the interleavings should be considered. The process $\mathsf{in}(c, x).P$ waits to receive a message on the public channel c, and then continues as indicated in P. However, the occurrence of the variable x in P will be replaced by the received message. The process $\mathsf{out}(c, u).P$ outputs the message u on the public channel c, and then continues as P. The process $!P$ represents an infinite number of copies of P running in parallel, i.e. $P \mid \ldots \mid P$. 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 $\mathsf{if}\ u_1 = u_2\ \mathsf{then}\ P\ \mathsf{else}\ Q$ first checks whether u_1 is equal to u_2 (modulo the equational theory), and runs P if equality holds or runs Q otherwise. Note that the terms u, u_1, and u_2 that occur in the grammar may contain variables. However, these variables will be instantiated during the execution, and these terms will become ground when the evaluation will take place.

The constructions $\mathsf{new}\, n.P$ 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.

▶ **Example 4.** To illustrate our syntax, we consider the RFID protocol described in Sec-

tion 2.1. Using our formalism, the two roles of this protocol are modelled as follows:

$$P_{\mathsf{tag}} = \mathsf{in}(c_T, x).\ \mathsf{new}\, n_T.\ \mathsf{out}(c_T, \langle \mathsf{id} \oplus n_T,\ \mathsf{h}(\langle x, k\rangle) \oplus n_T\rangle).\ 0$$

$$P_{\mathsf{reader}} = \mathsf{new}\, n_R.\ \mathsf{out}(c_R, n_R).\ \mathsf{in}(c_R, y).\ \mathsf{if}\ (\mathsf{proj}_1(y) \oplus \mathsf{id}) \oplus \mathsf{proj}_2(y) = \mathsf{h}(\langle n_R, , \mathsf{k}\rangle)\ \mathsf{then}\, 0\, \mathsf{else}\, 0.$$

where $c_T, c_R \in \mathcal{C}h$, $\mathsf{id} \in \mathcal{N}$, and $x, y \in \mathcal{X}$.

Then, we may consider the process $\mathsf{new}\ \mathsf{k}.\,\mathsf{new}\ \mathsf{id}.(P_{\mathsf{tag}} \mid P_{\mathsf{reader}})$ which corresponds to one instance of each role. We may also consider more complex scenario. For instance, the process $\mathsf{new}\ \mathsf{k}.\,\mathsf{new}\ \mathsf{id}.!\ (P_{\mathsf{tag}} \mid P_{\mathsf{reader}})$ represents multiple instances of the tag id (with key k) and multiple instances of a reader ready to communicate with tag id. Lastly, the process $!\,\mathsf{new}\ \mathsf{k}.\,\mathsf{new}\ \mathsf{id}.!\ (P_{\mathsf{tag}} \mid P_{\mathsf{reader}})$ represents a situation with many tags (and readers), each of them being able to execute many instances of the protocol.

Configurations represent processes together with a frame representing the knowledge of the attacker so far.

▶ **Definition 5.** A *configuration* is a pair $(\mathcal{P}; \phi)$ where

- $\mathcal{P}$ is a multiset of *ground processes*; and
- $\phi = \{\mathsf{w}_1 \mapsto u_1, \ldots, \mathsf{w}_n \mapsto u_n\}$ is a *frame*.

The applied-pi calculus, as introduced in [2], does not consider this notion of configurations but rely instead on a notion of extended processes and a notion of structural equivalence to identify processes that are equal modulo e.g. associativity and commutativity of the parallel operator. Our notion of configurations, also used in some other works, can be seen as a more canonical way to represent an extended process.

Then, we define the operational semantics of our calculus through a labelled transition system over configurations explaining how a process can evolve (see Figure 4). All the rules are rather standard and correspond to the informal semantics introduced at the beginning of this section. For instance, when outputting a message u on the public channel c, the resulting message is stored into the frame ϕ and is given to the attacker through the handle w_i. The rule In is more involved. The idea is that the attacker can build any term using his current knowledge and then send the resulting message to the participant. Therefore, the participant is ready to accept any term deducible by the attacker form ϕ. The rules Then and Else allow one to execute a conditional. Note that the equality is done modulo E. The three remaining rules allow one to execute a restriction, split a parallel composition, and unfold a replication. The In and Out rules are the only observable actions.

▶ **Example 6.** To continue with our running example, we consider $\mathcal{P}_{\mathsf{same}} = \{P_{\mathsf{tag}}, P_{\mathsf{tag}}\}$, and $\mathcal{P}_{\mathsf{diff}} = \{P_{\mathsf{tag}}, P'_{\mathsf{tag}}\}$ where P'_{tag} is as P_{tag} but id and k have been replaced by id' and k'. Let $\phi_0 = \{\mathsf{w}_1 \mapsto \mathsf{id}, \mathsf{w}_2 \mapsto \mathsf{id}'\}$. We have that:

$$\begin{array}{rl}
 & (\{P_{\mathsf{diff}}\}; \phi_0) \\
\xrightarrow{\mathsf{in}(c_T, n_R^0)}\xrightarrow{\tau} & (\{\mathsf{out}(c_T, \langle \mathsf{id} \oplus n_T, \mathsf{h}(\langle n_R^0, \mathsf{k}\rangle) \oplus n_T\rangle), P'_{\mathsf{tag}}\}; \phi_0) \\
\xrightarrow{\mathsf{out}(c_T, \mathsf{w}_3)} & (\{P'_{\mathsf{tag}}\}; \phi_0 \uplus \{\mathsf{w}_3 \mapsto \langle \mathsf{id} \oplus n_T, \mathsf{h}(\langle n_R^0, \mathsf{k}\rangle) \oplus n_T\rangle\}) \\
\xrightarrow{\mathsf{in}(c_T, n_R^0)}\xrightarrow{\tau} & (\{\mathsf{out}(c_T, \langle \mathsf{id}' \oplus n'_T, \mathsf{h}(\langle n_R^0, \mathsf{k}'\rangle) \oplus n'_T\rangle).0\}, \phi_0 \uplus \{\mathsf{w}_3 \mapsto \langle \mathsf{id} \oplus n_T, \mathsf{h}(\langle n_R^0, k\rangle) \oplus n_T\rangle\}) \\
\xrightarrow{\mathsf{out}(c_T, \mathsf{w}_4)} & (\{0\}, \phi_0 \uplus \{\mathsf{w}_3 \mapsto \langle \mathsf{id} \oplus n_T, \mathsf{h}(\langle n_R^0, \mathsf{k}\rangle) \oplus n_T\rangle,\ \mathsf{w}_4 \mapsto \langle \mathsf{id}' \oplus n'_T, \mathsf{h}(\langle n_R^0, \mathsf{k}'\rangle) \oplus n'_T\rangle\})
\end{array}$$

We denote ϕ_{diff} the resulting frame. The same sequence of actions is also feasible starting from $(\mathcal{P}_{\mathsf{same}}; \phi_0)$. Indeed, we have that:

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

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

$$\textsc{In}\quad (\{\mathsf{in}(c,z).P\} \uplus \mathcal{P}; \phi) \xrightarrow{\mathsf{in}(c,R)} (P\{z \mapsto u\} \uplus \mathcal{P}; \phi) \quad \text{when } \phi \vdash_{\mathsf{E}} u$$

$$\textsc{Out}\quad (\{\mathsf{out}(c,u).P\} \uplus \mathcal{P}; \phi) \xrightarrow{\mathsf{out}(c,w)} (P \uplus \mathcal{P}; \phi \cup \{\mathsf{w}_{i+1} \mapsto u\}) \quad \text{where } i = |dom(\phi)|.$$

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

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

$$\textsc{Repl}\quad (\{!P\} \uplus \mathcal{P}; \phi) \xrightarrow{\tau} (\{!P, P\} \uplus \mathcal{P}; \phi)$$

Figure 4 Semantics of our processes

$$(\mathcal{P}_{\mathsf{same}}; \phi_0) \xrightarrow{\mathsf{in}(c_T, n_R^0)\cdot\tau\cdot\mathsf{out}(c_T,\mathsf{w}_3)\cdot\mathsf{in}(c_T,n_R^0)\cdot\tau\cdot\mathsf{out}(c_T,\mathsf{w}_4)} (0; \phi_{\mathsf{same}})$$

where $\phi_{\mathsf{same}} = \phi_0 \uplus \{\mathsf{w}_3 \mapsto \langle \mathsf{id} \oplus n_T, \mathsf{h}(\langle n_R^0, \mathsf{k}\rangle) \oplus n_T\rangle,\ \mathsf{w}_4 \mapsto \langle \mathsf{id} \oplus n'_T, \mathsf{h}(\langle n_R^0, \mathsf{k}\rangle) \oplus n'_T\rangle\}$.

4 Modelling privacy-type properties

In order to express privacy-type properties such as the unlinkability property briefly discussed in Section 2, we need to formally define the notion of indistinguishability we are interested in. The notion of *trace equivalence* is formally introduced in Section 4.1, and then we explain how to express privacy-type security properties such as vote-privacy, unlinkability, or strong flavours of secrecy relying on this notion.

4.1 Trace equivalence

To start, we consider two protocols P and Q, and we assume a passive attacker who simply listens to the communication. We would like to know whether such a passive attacker is able (by simply listening to the communication) to tell which protocol is currently under execution: i.e. P or Q. Typically, the attacker will observe two sequences of messages, i.e. two frames, and he will try to distinguish them. Intuitively, two frames ϕ and ψ are in static equivalence if an attacker cannot distinguish them, i.e. any test that holds in ϕ also holds in ψ.

▶ **Definition 7.** Two frames ϕ and ψ are in *static equivalence*, written $\phi \sim_{\mathsf{E}} \psi$, if they have the same domain, i.e. $dom(\phi) = dom(\psi)$, and for any recipes R, $R' \in \mathcal{T}(\Sigma, dom(\phi))$, we have that: $R\phi =_{\mathsf{E}} R'\phi$ if, and only if, $R\psi =_{\mathsf{E}} R'\psi$.

▶ **Example 8.** Consider the two following frames:

- $\phi_{\mathsf{diff}} = \phi_0 \uplus \{\mathsf{w}_3 \mapsto \langle \mathsf{id} \oplus n_T, \mathsf{h}(\langle n_R^0, \mathsf{k}\rangle) \oplus n_T\rangle, \mathsf{w}_4 \mapsto \langle \mathsf{id}' \oplus n'_T, \mathsf{h}(\langle n_R^0, \mathsf{k}'\rangle) \oplus n'_T\rangle\}$, and
- $\phi_{\mathsf{same}} = \phi_0 \uplus \{\mathsf{w}_3 \mapsto \langle \mathsf{id} \oplus n_T, \mathsf{h}(\langle n_R^0, \mathsf{k}\rangle) \oplus n_T\rangle, \mathsf{w}_4 \mapsto \langle \mathsf{id} \oplus n'_T, \mathsf{h}(\langle n_R^0, \mathsf{k}\rangle) \oplus n'_T\rangle\}$.

The following test holds in ϕ_{same}: $\mathsf{proj}_1(\mathsf{w}_3) \oplus \mathsf{proj}_2(\mathsf{w}_3) \stackrel{?}{=} \mathsf{proj}_1(\mathsf{w}_4) \oplus \mathsf{proj}_2(\mathsf{w}_4)$.
Indeed, we have that:

- $[\mathsf{proj}_1(\mathsf{w}_3) \oplus \mathsf{proj}_2(\mathsf{w}_4)]\phi_{\mathsf{same}} =_{\mathsf{E}_{\mathsf{ex}}} (\mathsf{id} \oplus n_T) \oplus (\mathsf{h}(\langle n_R^0, \mathsf{k}\rangle) \oplus n_T) =_{\mathsf{E}_{\mathsf{ex}}} \mathsf{id} \oplus \mathsf{h}(\langle n_R^0, \mathsf{k}\rangle)$, and
- $[\mathsf{proj}_1(\mathsf{w}_3) \oplus \mathsf{proj}_2(\mathsf{w}_4)]\phi_{\mathsf{same}} =_{\mathsf{E}_{\mathsf{ex}}} (\mathsf{id} \oplus n'_T) \oplus (\mathsf{h}(\langle n_R^0, \mathsf{k}\rangle) \oplus n'_T) =_{\mathsf{E}_{\mathsf{ex}}} \mathsf{id} \oplus \mathsf{h}(\langle n_R^0, \mathsf{k}\rangle)$.

However, this test does not hold in ϕ_{diff} since $\mathsf{id} \neq \mathsf{id}'$ and $\mathsf{k} \neq \mathsf{k}'$. This means that an attacker can observe a difference between these two frames by xoring the two components of each message and checking whether this computation yields an equality, therefore retrieving the attack described in Section 2.1. Note that such an equality crucially relies on the algebraic properties of the exclusive-or operator.

Then, trace equivalence is the active counterpart of static equivalence taking into account the fact that the attacker may interfere during the execution of the process in order to distinguish between the two situations. Given a configuration $K = (\mathcal{P}; \phi)$, we define $\mathsf{trace}(K)$ as follows: $\mathsf{trace}(K) = \{(\mathsf{tr}, \phi') \mid (\mathcal{P}; \phi) \xrightarrow{\mathsf{tr}} (\mathcal{P}'; \phi') \text{ for some configuration } (\mathcal{P}'; \phi')\}$.

▶ **Definition 9.** Given two configurations K_P and K_Q, $K_P \sqsubseteq_t K_Q$ if for every $(\mathsf{tr}, \phi) \in \mathsf{trace}(K_P)$, there exists $(\mathsf{tr}', \psi) \in \mathsf{trace}(K_Q)$ such that tr and tr' are equal up to τ actions, and $\phi \sim_{\mathsf{E}} \psi$. The configuration K_P and K_Q are in *trace equivalence*, denoted $K_P \approx_t K_Q$, if $K_P \sqsubseteq_t K_Q$ and $K_Q \sqsubseteq_t K_P$.

4.2 Some security properties

We show here how the notion of trace equivalence can be used to model interesting privacy-type properties.

Unlinkability

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. Going back to our BAC protocol, this can be expressed through the following equivalence:

$$\begin{array}{c} !\ \mathsf{new\ ke.\ new\ km.}\ !\,(P_{\mathsf{tag}}(\mathsf{ke},\mathsf{km}) \mid P_{\mathsf{reader}}(\mathsf{ke},\mathsf{km})) \\ \approx_t \\ !\ \mathsf{new\ ke.\ new\ km.}(P_{\mathsf{tag}}(\mathsf{ke},\mathsf{km}) \mid P_{\mathsf{reader}}(\mathsf{ke},\mathsf{km})). \end{array}$$

In case of the french implementation of the BAC protocol, as explained in Section 2.2, it has been shown that this equivalence does not hold [4].

▶ **Example 10.** To illustrate our notion of trace equivalence, we consider the RFID protocol given in Section 2.1. In order to simplify the setting, we consider a simple scenario which consists of two sessions of the role of the tag. We assume that one session is executed by the tag with identity id and key k, whereas the second one is executed either by the same tag or another one. We would like to know whether the attacker is able to distinguish these two situations. This corresponds to the configurations $K_{\mathsf{same}} = (\mathcal{P}_{\mathsf{same}}; \phi_0)$ and $K_{\mathsf{diff}} = (\mathcal{P}_{\mathsf{diff}}; \phi_0)$ described in Example 6. Actually, we have that K_{same} and K_{diff} are not in trace equivalence. More precisely, we have that $K_{\mathsf{same}} \not\sqsubseteq_t K_{\mathsf{diff}}$. Indeed, we have shown that:

- $K_{\mathsf{same}} \xrightarrow{\mathsf{in}(c_T, n_R^0) \cdot \tau \cdot \mathsf{out}(c, \mathsf{w}_3) \cdot \mathsf{in}(c_T, n_R^0) \cdot \tau \cdot \mathsf{out}(c_T, \mathsf{w}_4)} (0, \phi_{\mathsf{same}})$ (see Example 6); and
- $[\mathsf{proj}_1(\mathsf{w}_3) \oplus \mathsf{proj}_2(\mathsf{w}_3) =_{\mathsf{E}_{\mathsf{ex}}} \mathsf{proj}_1(\mathsf{w}_4) \oplus \mathsf{proj}_2(\mathsf{w}_4)]\phi_{\mathsf{same}}$ (see Example 8).

However, the only configuration $(\mathcal{P}', \phi')$ such that (up to some τ actions)

$$K_{\mathsf{diff}} \xrightarrow{\mathsf{in}(c,n_R^0)\cdot\mathsf{out}(c,\mathsf{w}_3)\cdot\mathsf{in}(c,n_R^0)\cdot\mathsf{out}(c,\mathsf{w}_4)} (P', \phi')$$

is $(0, \phi_{\mathsf{diff}})$ and we have seen that $\mathsf{proj}_1(\mathsf{w}_3) \oplus \mathsf{proj}_2(\mathsf{w}_3) \stackrel{?}{=} \mathsf{proj}_1(\mathsf{w}_4) \oplus \mathsf{proj}_2(\mathsf{w}_4)$ does not hold in ϕ_{diff} (see Example 8).

This corresponds to the attack scenario briefly described in Section 2.1.

Vote secrecy

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 secrecy is typically defined (see e.g. [44]) by the fact that an observer should not observe when two honest 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 swapped their vote. This security property is formally expressed as follows:

$$S[V(A, yes) \mid V(B, no)] \approx_t S[V(A, no) \mid V(B, yes)].$$

Ideally, such an equivalence should be established considering an empty context S. However, very often such a property holds under some trusted assumptions. For instance, we may have to trust the tallying authority. The context S makes explicit all these assumptions.

Strong flavours of secrecy

The notion of secrecy usually considered by symbolic approaches is a weak form of secrecy expressed as a non-deducibility property. However, relying on trace equivalence, we can also express strong forms of secrecy. Intuitively, strong secrecy means that an attacker cannot see any difference when the value of the secret changes [14]. One way to express this it to let the attacker choose the values of the secret:

$$\mathsf{in}(c, x_1).\mathsf{in}(c, x_2).P(x_1) \approx_t \mathsf{in}(c, x_1).\mathsf{in}(c, x_2).P(x_2).$$

Intuitively, in the equivalence above, $P(x)$ is the protocol in which the value of the secret is replaced by x, i.e. by a value chosen by the attacker. Another flavour of secrecy of interest is *real-or-random secrecy*. The idea is to let the attacker interact with the protocol, and once the secret value has been established, typically a fresh session key k, we want to see if the attacker is able to distinguish the real situation (the one in which the fresh established key k will be used) from an ideal situation in which the key k is replaced by fresh random value r. If the adversary is unable to distinguish these two scenarios, we say that the protocol satisfies real-or-random secrecy of the secret key.

The notion of trace equivalence can also be used in presence of low entropy secret such as the values ke and km in the BAC protocol to check resistance of the protocol to off-line guessing attacks. This can be modelled relying on trace equivalence by checking whether the attacker can see the difference between a scenario where the real password is revealed at the end, and another one where a wrong password is revealed (see e.g. [36]).

5 Verifying equivalence-based properties

The formal symbolic approach allows one to benefit from existing verification tool that rely on various techniques ranging from model-checking to resolution, and rewriting techniques.

This is appealing as manual proofs are tedious and error-prone. However, verifying in such a setting the most simple form of secrecy (expressed as a non-deducibility of a term) is a difficult problem which is well-known to be undecidable. Privacy-type properties are actually more difficult to handle and have been shown undecidable even for some classes where secrecy is actually decidable [32].

5.1 Warm-up

Several papers are devoted to the study of the intruder deduction problem, i.e. the problem of deciding whether a term (typically the secret) is deducible from a given set of terms representing the knowledge of the attacker. This problem has been shown decidable (often in PTIME) for various equational theories, e.g. homomorphic encryption, blind signatures, various equational theories with an associative and commutative symbol (AC). However, when considering equivalence-based properties, the natural question we have to solve is not to decide whether a term is deducible or not, but rather whether two frames are indistinguishable or not. This problem can be formally stated as follows:

Static equivalence problem:
Input Two frames ϕ and ψ having the same domain.
Output Are ϕ and ψ in static equivalence, i.e. $\phi \sim_{\mathsf{E}} \psi$?

Again depending on the equational theory under study, this problem may or may not be decidable. Actually, even if such a problem is often more complex to solve than the intruder deduction problem, this problem is now well-understood. Efficient (often PTIME) algorithms and tools (e.g. FAST [35], and KISS [33]) have been developed to solve this problem for various equational theories. Some existing results for deduction and static equivalence are briefly summarised in Figure 5. Moreover, thanks to the combination result provided in [39], deduction and static equivalence are also decidable for the union of any disjoint theories of this tabular.

Theory E	Deduction	Static Equivalence
subterm convergent	PTIME [1]	
blind sign., addition, homo. encryption	decidable [1]	
ACU	NP-complete	decidable [1] PTIME [39]
ACUN/AG	PTIME [27]	PTIME [1, 39]
ACUh	NP-complete [52]	decidable [39]
ACUNh/AGh	PTIME [43]	decidable [39]
$\mathsf{AGh}_1 \ldots \mathsf{h}_n$	decidable [39]	decidable [39]

Figure 5 Some decidability and complexity results for deduction and static equivalence.

5.2 Bounded number of sessions

When analysing a protocol, a reasonable assumption under which the verification problem has been shown decidable is the so-called bounded number of sessions assumption. This

amounts to consider processes without replication. Note that processes without replication allows us to consider traces of bounded length, but the problem remains difficult: the labelled transition system representing all the possible interactions of the honest participants with the attacker is still infinitely branching. 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.

Some theoretical results. Under such an assumption, the problem of deciding trace equivalence has been first shown decidable in [50], where a fragment of the spi-calculus (no replication, no else branch) is considered. In 2005, Baudet designs 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 [13]. Few years later, a shorter proof of this result was proposed by Chevalier and Rusinowitch in [28]. In this work, it is shown that when two processes are not in trace equivalence, then there exists a small witness of this fact. The main issue with the results mentioned so far is practicality. Consequently, they have not been implemented.

Since then, a lot of progress has been made leading to more efficient procedures implemented in various verification tools. We review the most popular ones and briefly explain their features.

Spec. This tools implements a decision procedure for the notion of *open bisimulation*: a notion that is strictly stronger than trace equivalence [60]. Processes given in input are written in the spi-calculus syntax and else branches are not allowed. Regarding the cryptographic primitives, the tool has been recently extended to deal with asymmetric primitives, and therefore is now able to handle all the standard primitives.

Akiss. The procedure described in [20] deals with rich user-defined term algebras provided that they can be defined using a convergent rewriting system enjoying the *finite variant property* [34].This includes all the standard primitives, but also some other primitives such blind signatures, and trapdoor commitment used e.g. in electronic voting protocols. However, due to some approximations, this procedure is only able to check trace equivalence for the class of determinate processes. Moreover, its termination is only guaranteed for subterm convergent equational theories. Regarding the input syntax, processes are written as linear roles and originally the tool only allows inputs, outputs, and equality tests. Recently, some extensions have been implemented. In particular, the procedure has been extended to deal with the exclusive-or operator [8], and various RFID protocols have been analysed such as the RFID protocol presented in Section 2.1. The tool has also been extended to deal with else branches [47].

Apte/DeepSec. The tool Apte [21] implements the decision procedure described in [23]. Such a procedure deals with all the standard cryptographic primitives. Actually, the procedure presented in [23] allows for slightly more general processes than those presented in Section 3 since it deals with private channels and internal communications. This procedure has been extended to deal with some forms of *side-channel* attacks regarding the length of messages [25], and the computation time [24]. Recently, getting some inspiration from the Apte tool, a new verification tool DeepSec has been implemented [26]. It deals with a large variety of cryptographic primitives that encompasses all the standard primitives. Moreover, it is significantly more efficient than other existing tools, namely Spec, Akiss, and its predecessor Apte.

SAT-Equiv. Following an approach originally developed for checking reachability properties [5], SAT-Equiv relies on more general verification techniques, namely graph planning and SAT-solving [38]. The procedure deals with symmetric encryption and pairs, and only consider simple processes (each process in parallel works on a dedicated channel) without else branches. However, an extension is currently under preparation and the tool will be able to cover all standard primitives soon. In order to benefit from graph planning and SAT-solvers, the size of messages has to be bounded and this bound needs to be practical. The soundness of the tool is based on a typing result [30] guaranteeing the existence of a witness of non-equivalence where messages comply to a certain format (induced by a typing system). The resulting implementation, SAT-Equiv, outperforms other existing tools. It can analyse several sessions (typically more than 20 for rather simple protocols) where most existing tools have to stop after few sessions. Termination has been established, and this is the most efficient tool able to decide trace equivalence. However, the class of processes that it is able to consider is rather limited (e.g. no else branch, simple processes satisfying a type-compliance assumption).

5.3 Unbounded number of sessions

The decidability results and the tools mentioned so far only consider a bounded number of sessions, and thus assume that the protocol is executed a limited number of times. The problem is that even if the protocol has been proved secure for n sessions, there is no guarantee that the protocol will remain secure if it is executed one more time.

Some theoretical results. It is well-known that replication allowing us to model an unbounded number of sessions leads to undecidability even when considering a simple secrecy property. The first decidability result for trace equivalence has been obtained for a rather restricted class: the class of *ping-pong protocols* built using standard primitives (but without pairs) [32]. This result has been obtained through a characterisation of equivalence of protocols in terms of equality of languages of (generalised, real-time) deterministic pushdown automata.

Assuming finitely many nonces and keys, another decidability result has been obtained in [30] for the class of simple processes built using symmetric encryptions and pairs. Such a decidability result is based on a typing result which means that messages comply to a certain format. A well-known class of protocols that satisfies such a requirement, is the class of *tagged protocols*. The typing result mentioned above has also been used to establish the first decidability result in presence of fresh nonces [31]. This decidability result inherits the restrictions of the typing result (symmetric encryption only, type-compliance) on which it is based. Additionally, a notion of dependency graph allowing one to represent the dependencies between the actions of the protocols is carefully designed. In case this graph is acyclic, a bound on the length of an attack trace can be deduced, giving us an algorithm to decide trace equivalence.

ProVerif, Tamarin, and Maude-NPA. Since the problem of checking trace equivalence for rich class of protocols is undecidable, many works aim at developing procedures that are sound but not complete w.r.t. trace equivalence. In particular, several tools consider the notion of *diff-equivalence* (a notion stronger than trace equivalence). This notion has been introduced in [16] and implemented in the ProVerif tool. Since then, this notion of diff-equivalence has been integrated in Tamarin [12] and Maude-NPA [57]. Due to the fact that the equivalence under study is the so-called notion of diff-equivalence, these tools are not well-suited to analyse some privacy-type properties such as unlinkability, or vote secrecy.

To extend the scope of the ProVerif tool, several extensions have been recently proposed to go beyond diff-equivalence, e.g. [22, 17]. For instance, ProSwapper [17] allows one to go beyond diff-equivalence by rearranging automatically processes before launching ProVerif. This front end is particularly relevant to analyse vote secrecy. ProVerif has also been used as a back end to analyse anonymity and unlinkability properties [48]. This approach proposes sufficient conditions that are actually checkable using ProVerif, and from which the security of the protocol can be established. This method allows to automatically verify unlinkability and anonymity of some protocols that were out of the scope of existing tools, e.g. unlinkability of the fixed version of the BAC protocol has been established for the first time relying on this technique, and some of the weaknesses presented in Section 2.2 on the PACE protocol have been discovered using this method.

Whereas ProVerif and Maude-NPA are completely automatic, Tamarin provides two ways of constructing proofs: an efficient, fully automated mode that uses heuristics to guide proof search and an interactive mode. Regarding the cryptographic primitives, these tools support a rich term algebra including all the standard primitives. In addition, Tamarin also supports Diffie-Hellman exponentiation, and recently exclusive-or has been added into the tool. The Maude-NPA tool also supports a rich term algebra but the tool suffers from termination issues, especially when considering the exclusive-or operator. Despite some non-termination issues that may happen from time to time, these tools are efficient. For instance, ProVerif generally concludes within few seconds. These good performances are due to some well-chosen over-approximations that are done on the protocols at the beginning of the security analysis that may lead sometimes to false attacks.

Type-Eq. Recently, an approach based on type systems has been developed and implemented in the tool Type-Eq [40, 41]. This approach is very efficient and can prove security of protocols that require a mix of bounded and unbounded number of sessions This tool only consider the standard cryptographic primitives, and requires the user to enter all the information regarding types. While this approach allows to go beyond diff-equivalence, e.g. allowing else branches to be matched with then branches, it is not yet possible to analyse e.g. unlinkability of the BAC protocol.

6 Some challenges

In the past ten years, equivalence-based properties have received a lot of attention and we now have tools to check automatically privacy-type security properties when considering rather simple protocols. However, new applications are coming or are already there (electronic voting, contactless payment, keyless systems, ...) and these applications often rely on security protocols that can not be analysed relying on existing verification tools due to various reasons.

State-explosion problem. Systems we are interested in are highly concurrent and all the existing methods and tools which naively explore all possible symbolic interleavings are causing the so called state-explosion problem. This problem seriously limits the practical impact of tools such as Akiss, Spec, Apte and, to a lesser extent, DeepSec. Actually, recent works [9, 10] have partially addressed this issue by developing dedicated partial order reduction (POR) techniques to dramatically reduce the number of interleavings to explore. They have been implemented in Apte, Akiss, and DeepSec, and brought significant speed-up. However, these techniques can only be applied on action-deterministic processes, and this is not sufficient to analyse e.g. unlinkability. To mitigate this problem, it should be possible to leverage classical POR techniques for use in the specific security setting. A recent result

in this direction has been obtained in [11] regarding the persistent and sleep sets techniques, and other POR techniques deserve attention to obtain performance gains.

Cryptographic primitives. Most of the tools deal with standard primitives, i.e. encryptions, signatures, and hashes. However, many protocols, such as RFID protocols or electronic voting protocols rely on primitives that do not fall into this class. For instance, protocols that contain some time-critical steps often rely on low-level operator to reduce computation (and communication) time. As demonstrated by some recent works (e.g. [8]), dealing with a simple operator such as the exclusive-or and its algebraic properties in the symbolic setting is actually challenging. Electronic voting protocols have to achieve antagonist security properties and they often rely on exotic cryptographic primitives to try to achieve them, e.g. blind signatures, zero-knowledge proofs, homomorphic encryptions, ... To avoid missing attacks, these primitives together with their algebraic properties have to be modelled faithfully when performing the formal security analysis.

Mutable global states. Many modern protocols involve a notion of state meaning that some data are conveyed from one session to another through e.g. a register. This is the case for instance of many RFID protocols as those presented in [18, 61]. Several protocols proposed by the 3rd Generation Partnership Project (3GPP) as a standard for 3G and 4G mobile network communications are also stateful. For instance, the Authentication and Key Agreement (AKA) protocol relies on a state to store a counter across different sessions, and a state is also used in a crucial way to store temporary identifiers (namely TMSI) in the TMSI reallocation procedure. Moreover, these protocols are supposed to guarantee unlinkability. Existing results regarding the formal verification of such protocols model states in a very abstract way, considering for instance that the client and the server already (magically) share a fresh name instead of modelling the sequence number mechanism. Recent advances have been made in this direction though an extension of the Tamarin prover [46]. Nevertheless, methods for checking trace equivalence on stateful protocols are in their infancy.

References

1 M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, 2006.

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

3 M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. of the 4th ACM conference on Computer and communications security*, pages 36–47. ACM, 1997.

4 M. Arapinis, T. Chothia, E. Ritter, and M. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Computer Society Press, 2010.

5 A. Armando, R. Carbone, and L. Compagna. SATMC: a SAT-based model checker for security-critical systems. In *Proc. 20th international Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14)*, pages 31–45. Springer, 2014.

6 A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. T. Abad. Formal analysis of saml 2.0 web browser single sign-on: Breaking the saml-based single sign-on for google apps. In *Proc. 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*, pages 1–10, 2008.

7 A. Armando et al. The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures. In *Proc. 18th International Conference on Tools*

and Algorithms for the Construction and Analysis of Systems (TACAS'12), volume 7214, pages 267–282. Springer, 2012.

8 D. Baelde, S. Delaune, I. Gazeau, and S. Kremer. Symbolic verification of privacy-type properties for security protocols with xor. In *Proc. 30th IEEE Computer Security Foundations Symposium (CSF'17)*, pages 234–248. IEEE Computer Society Press, 2017.

9 D. Baelde, S. Delaune, and L. Hirschi. Partial order reduction for security protocols. In *Proc. 26th International Conference on Concurrency Theory (CONCUR'15)*, volume 42 of *LIPIcs*, pages 497–510. Leibniz-Zentrum für Informatik, 2015.

10 D. Baelde, S. Delaune, and L. Hirschi. A reduced semantics for deciding trace equivalence. *Logical Methods in Computer Science*, 2017.

11 D. Baelde, S. Delaune, and L. Hirschi. POR for Security Protocols Equivalences - Beyond Action-Determinism. Technical report, 2018.

12 D. Basin, J. Dreier, and R. Sasse. Automated symbolic proofs of observational equivalence. In *Proc. 22nd Conference on Computer and Communications Security (CCS'15)*, pages 1144–1155. ACM, 2015.

13 M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proc. 12th ACM conference on Computer and communications security (CCS'05)*, pages 16–25. ACM, 2005.

14 B. Blanchet. Automatic proof of strong secrecy for security protocols. In *Proc. . 2004 Symposium on Security and Privacy*, pages 86–100. IEEE Computer Society Press, 2004.

15 B. Blanchet. An automatic security protocol verifier based on resolution theorem proving (invited tutorial). In *Proc. 20th International Conference on Automated Deduction (CADE-20)*, July 2005.

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

17 B. Blanchet and B. Smyth. Automated reasoning for equivalences in the applied pi calculus with barriers. In *Proc. 29th Computer Security Foundations Symposium (CSF'16)*, 2016.

18 M. Brusó, K. Chatzikokolakis, and J. Den Hartog. Formal verification of privacy for RFID systems. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*, pages 75–88. IEEE Computer Society Press, 2010.

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

20 R. Chadha, Ş. Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. In *Proc. European Symposium on Programming (ESOP'12)*, pages 108–127. Springer, 2012.

21 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)*, volume 8413 of *LNCS*, pages 587–592, 2014.

22 V. Cheval and B. Blanchet. Proving more observational equivalences with ProVerif. In *Proc. 2nd Conference on Principles of Security and Trust (POST'13)*, volume 7796 of *LNCS*, pages 226–246. Springer, 2013.

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

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

25 V. Cheval, V. Cortier, and 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)*, volume 8044 of *LNCS*, pages 708–723. Springer Berlin Heidelberg, 2013.

26 V. Cheval, S. Kremer, and I. Rakotonirina. Deepsec: Deciding equivalence properties in security protocols - theory and practice. In *Proc. 39th IEEE Symposium on Security and Privacy (S&P'18)*. IEEE Computer Society Press, May 2018. Accepted for publication.

27 Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and product in exponents. In *Proc. 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'03)*, volume 2914 of *LNCS*, pages 124–135. Springer-Verlag, 2003.

28 Y. Chevalier and M. Rusinowitch. Decidability of symbolic equivalence of derivations. *Journal of Automated Reasoning*, 48(2):263–292, 2012.

29 T. Chothia and V. Smirnov. A traceability attack against e-passports. In *Proc. 14th International Conference on Financial Cryptography and Data Security (FC'10)*, 2010.

30 R. Chrétien, V. Cortier, and S. Delaune. Typing messages for free in security protocols: the case of equivalence properties. In *Proc. 25th International Conference on Concurrency Theory (CONCUR'14)*, volume 8704 of *LNCS*, pages 372–386. Springer, 2014.

31 R. Chrétien, V. Cortier, and S. Delaune. Decidability of trace equivalence for protocols with nonces. In *Proc. 28th Computer Security Foundations Symposium (CSF'15)*, pages 170–184. IEEE Computer Society Press, 2015.

32 R. Chrétien, V. Cortier, and S. Delaune. From security protocols to pushdown automata. *ACM Transactions on Computational Logic*, 17(1:3), Sept. 2015.

33 Ş. Ciobâcă, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. *Journal of Automated Reasoning*, 48(2):219–262, Feb. 2012.

34 H. Comon-Lundh and 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)*, pages 294–307. Springer, 2005.

35 B. Conchinha, D. A. Basin, and C. Caleiro. FAST: an efficient decision procedure for deduction and static equivalence. In *Proc. 22nd International Conference on Rewriting Techniques and Applications, RTA 2011*, volume 10 of *LIPIcs*, pages 11–20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

36 R. Corin, J. Doumen, and S. Etalle. Analysing password protocol security against off-line dictionary attacks. *Electonic Notes in Theoretical Computer Science*, 121:47–63, 2005.

37 V. Cortier. *Vérification automatique des protocoles cryptographiques*. Thèse de doctorat (PhD thesis), Laboratoire Spécification et Vérification, ENS Cachan, France, Mar. 2003.

38 V. Cortier, A. Dallon, and S. Delaune. Sat-equiv: an efficient tool for equivalence properties. In *Proc. 30th IEEE Computer Security Foundations Symposium (CSF'17)*. IEEE Computer Society Press, Aug. 2017.

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

40 V. Cortier, N. Grimm, J. Lallemand, and M. Maffei. A type system for privacy properties. In *24th ACM Conference on Computer and Communications Security (CCS'17)*, pages 409–423. ACM, October 2017.

41 V. Cortier, N. Grimm, J. Lallemand, and M. Maffei. Equivalence properties by typing in cryptographic branching protocols. In *Proc. 7th International Conference on Principles of Security and Trust (POST'18)*, pages 160–187, April 2018.

42 V. Cortier and B. Smyth. Attacking and fixing helios: An analysis of ballot secrecy. In *Proc. 24th Computer Security Foundations Symposium (CSF'11)*, pages 297–311. IEEE Computer Society Press, 2011.

43 S. Delaune. Easy intruder deduction problems with homomorphisms. *Information Processing Letters*, 97(6):213–218, Mar. 2006.

44 S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi calculus. *Journal of Computer Security*, 18(2):317–377, Mar. 2010.

45 D. Dolev and A. C. Yao. On the security of public key protocols. In *Proc. 22nd Symposium on Foundations of Computer Science (FCS'81)*, pages 350–357. IEEE Computer Society Press, 1981.

46 J. Dreier, L. Hirschi, S. Radomirovic, and S. Ralf. Automated unbounded verification of stateful cryptographic protocols with exclusive OR operations. In *Proc. 31st IEEE Computer Security Foundations Symposium (CSF'18)*. IEEE Computer Society Press, 2018.

47 I. Gazeau and S. Kremer. Automated analysis of equivalence properties for security protocols using else branches. In *Proc. 22nd European Symposium on Research in Computer Security (ESORICS'17)*, volume 10493 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Sept. 2017.

48 L. Hirschi, D. Baelde, and S. Delaune. A method for verifying privacy-type properties: the unbounded case. In *Proc. 37th Symposium on Security and Privacy (S&P'16)*, 2016.

49 C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978.

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

51 I. Kim, E. Y. Choi, and D. H. Lee. Secure mobile RFID system against privacy and security problems. In *Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, SECPerU 2007, Istanbul, Turkey, July 19, 2007*, pages 67–72. IEEE Computer Society, 2007.

52 P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for AC-like equational theories with homomorphisms. In *Proc. 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *LNCS*, pages 308–322. Springer, 2005.

53 G. Lowe. An attack on the Needham-Schroeder public key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.

54 R. Milner. *Communicating and mobile systems - the Pi-calculus*. Cambridge University Press, 1999.

55 R. Needham and M. Schroeder. Using encryption for authentification in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

56 Technical advisory group on machine readable travel documents (tag/mrtd).

57 S. Santiago, S. Escobar, C. Meadows, and J. Meseguer. A formal definition of protocol indistinguishability and its verification using Maude-NPA. In *Security and Trust Management*, pages 162–177. Springer, 2014.

58 B. Schmidt, S. Meier, C. J. F. Cremers, and D. Basin. Automated analysis of Diffie-Hellman protocols and advanced security properties. In *Proc. 25th Computer Security Foundations Symposium (CSF'12)*, pages 78–94. IEEE Computer Society Press, 2012.

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

60 A. Tiu. A trace based bisimulation for the spi calculus. In *Programming Languages and Systems*, pages 367–382. Springer, 2007.

61 T. van Deursen and S. Radomirovic. Attacks on RFID protocols. *IACR Cryptology ePrint Archive - Report 2008/310*, 2008.