APTE:
an Algorithm for Proving Trace Equivalence

Vincent Cheval

School of Computer Science, University of Birmingham

Abstract. This paper presents APTE, a new tool for automatically proving the security of cryptographic protocols. It focuses on proving trace equivalence between processes, which is crucial for specifying privacy type properties such as anonymity and unlinkability.
The tool can handle protocols expressed in a calculus similar to the applied-pi calculus, which allows us to capture most existing protocols that rely on classical cryptographic primitives. In particular, APTE handles private channels and else branches in protocols with bounded number of sessions. Unlike most equivalence verifier tools, APTE is guaranteed to terminate.
Moreover, APTE is the only tool that extends the usual notion of trace equivalence by considering "side-channel" information leaked to the attacker such as the length of messages and the execution times. We illustrate APTE on different case studies which allowed us to automatically (re)-discover attacks on protocols such as the *Private Authentication* protocol or the protocols of the electronic passports.

1 Introduction

Cryptographic protocols are small distributed programs specifically designed to ensure the security of our communications on public channels like Internet. It is therefore essential to verify and prove the correctness of these cryptographic protocols. Symbolic models have proved their usefulness for verifying cryptographic protocols. However, the many sources of unboundedness in modelling of the capabilities of an attacker makes it extremely difficult to verify the security properties of a cryptographic protocol by hand. Thus, developing automatic verification tools of security protocols is a necessity. Since the 1980s, many tools have been developed to automatically verify cryptographic protocols (e.g. Scyther [10], ProVerif [3], Avispa [13] and others) but they mainly focus on *trace properties* such as authentication and secrecy, which typically specify that a protocol cannot reach a bad state. However, many interesting security properties such as anonymity, unlinkability, privacy, cannot be expressed as a trace property but require the notion of *equivalence property*. Intuitively, these properties specify the indistinguishability of some instances of the protocols. We focus here on the notion of *trace equivalence* which is well-suited for the analysis of security protocols.

Existing Tools. To our knowledge, there are only three tools that can handle equivalence properties: ProVerif [3], SPEC [12] and AKiSs [9]. The tool ProVerif originally was designed to prove trace properties but it can also check some equivalence properties (so-called diff-equivalence) [4] that are usually too strong to model a real intruder since they consider that the intruder has complete knowledge of the internal states of all the honest protocols executed. Note that this is the only tool that can handle an unbounded number of sessions of a protocol with a large class of cryptographic primitives in practice. However, the downside of ProVerif is that it may not terminate and it may also return a false-negative. More recently, the tool AKiSs [9] was developed in order to decide the trace equivalence of bounded processes that do not contain non-trivial else branching. This tool was proved to be sound and complete and accepts a large class of primitives but the algorithm was only conjectured to terminate. At last, the tool SPEC [12] is based on a decision procedure for open-bisimulation for bounded processes. The scope is however limited: open-bisimulation coincides with trace equivalence only for determinate processes and the procedure also assumes a fixed set of primitives (symmetric encryption and pairing), and a pattern based message passing, hence, in particular, no non-trivial else branching.

Hence, some interesting protocols cannot be handle by these tools. It is particularly the case for the Private Authentication protocol [2], or the protocols of the electronic passport [1] since they rely on a conditional with a non-trivial else branch to be properly modelled. Moreover, even though recent work [6] led to a new release of ProVerif that can deal with the Private Authentication protocol, ProVerif is still not able to handle the protocols of the electronic passport and yields a false positive due to the too strong equivalence that it proves.

At last none of the existing tools take into account the fact that an attacker can always observe the length of a message, even though it can leak information on private data. For example, in most of existing encryption schemes, the length of ciphertext depends on the length of its plaintext. Thus the ciphertext $\{m\}_k$ corresponding to the encryption of a message m by the key k can always be distinguished from the ciphertext $\{m, m\}_k$ corresponding to the encryption of the message m repeated twice by the key k. This is simply due to the fact that $\{m, m\}_k$ is longer than $\{m\}_k$. However, these two messages would be considered as indistinguishable in all previous mentioned tools.

2 Trace equivalence

Our tool is based on a symbolic model where the messages exchanged over the network are represented by *terms*. They are built from a set of variables and names by applying function symbols modelling the cryptographic primitives. For example, the symbol function senc (resp. sdec) represents the symmetric encryption (resp. decryption) primitive and the term $\mathsf{senc}(m, k)$ models the encryption of a message m by a key k. The behaviour of each primitive is modelled by a *rewriting system*. As such, a term $\mathsf{sdec}(\mathsf{senc}(m, k), k)$ will be rewritten m to model the fact that decrypting a ciphertext by the key that was used to en-

crypt indeed yields the plain text. Moreover, to take into account the length of messages, we associate to each cryptographic primitive f a length function $\ell_{\mathsf{f}} : \mathbb{N}^n \to \mathbb{N}$ where n is the arity of f. Intuitively, a length function represents the length of the outcome of a cryptographic primitive depending on the length of his inputs. For example, the length function for the pairing could be the function $\ell_{\langle\rangle}(x, y) = x + y + 1$. The length function $\ell_{\mathsf{senc}}(x, y) = x$ for the symmetric encryption would imply that the length of a ciphertext is always the size of its plaintext. Typically, each encryption scheme would have a specific length function that depends on the characteristics of the encryption scheme.

Equivalence of sequence of terms. By interacting with a protocol, an attacker may obtain a sequence of messages, meaning that not only he knows each message but also the order in which he obtained them. Properties like anonymity rely on the notion of indistinguishability between two sequences of messages. This is called *static equivalence* and denoted by $\sim$. For example, consider the two sequences of messages $\Phi_m = [k; \mathsf{senc}(m, k)]$ and $\Phi_n = [k; \mathsf{senc}(n, k)]$ where m, n are two random numbers. The two sequences Φ_n and Φ_m are indistinguishable. Indeed, even if the attacker can decrypt the second message using the first message, *i.e.* k, he obtains in both cases two random numbers and so does not gain any particular informations. However, the two sequences $[k; \mathsf{senc}(m, k); m]$ and $[k; \mathsf{senc}(m, k); n]$ are distinguishable since the attacker can compare the plain text of the ciphertext with the third message he obtained.

When the attacker can also compare the size of message, the equivalence is called *length static equivalence* and denoted $\sim_\ell$.

Processes. Participants in the protocol are modelled as processes whose grammar is as follows:

$$P \mid Q \qquad P + Q \qquad \mathsf{new}\ k; P \qquad \mathsf{out}(u, v); P \qquad \mathsf{in}(u, x); P$$
$$\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q \qquad \mathsf{let}\ x = u\ \mathsf{in}\ P$$

where P, Q are processes, u, v are terms and x is a variable. The nil process is denoted 0. The process $P+Q$ represents the non-deterministic choice between P and Q. The process $\mathsf{new}\ k$ is the creation of a fresh name. The process $\mathsf{out}(u, v)$ represents the emission of the message v into the channel u. Similarly, $\mathsf{in}(u, x)$ is the process that receives a message on the channel u and binds it to x. Typically, an attacker can interact with the process by emitting or receiving messages from honest participants through public channels. Hence we represent possible interactions of the attacker with P by the notion of *trace*, that is a pair (s, Φ) where s is the sequence of actions that the attacker performs and Φ is the sequence of messages that the attacker receives from the honest participants. A process is said to be determinate when the execution of the process is deterministic. For example, a process containing the choice operator is not determinate.

Definition 1 ((length) trace equivalence). *Let P and Q be two processes. The processes P and Q are trace equivalence if for all traces (s, Φ) of P there exists a trace (s', Φ') of Q such that $s = s'$ and $\Phi \sim \Phi'$ (and conversely).*

Moreover, we say that P and Q are in length trace equivalence when the length static equivalence $\sim_\ell$ is used to compare the sequence of messages Φ and Φ', i.e. $\Phi \sim_\ell \Phi'$.

Intuitively, this definition indicates that whatever the actions the attacker performs on P, the same actions can be performed on Q and the sequences of messages obtained in both cases are indistinguishable, and conversely.

3 The tool APTE

We present the tool APTE that decides the trace equivalence for bounded of (possibly non-determinate, possibly with non-trivial else branches) processes that use standard primitives, namely signature, pairing, symmetric and asymmetric encryptions and any one-way functions such as hash, mac, etc. Moreover, by specifying the linear length functions of each cryptographic primitives, a user can also use APTE to decide the length trace equivalence between processes. When the trace equivalence or length trace equivalence between two processes is not satisfied, APTE provides a witness of the non-equivalence, *i.e.* it displays the actions that the attacker has to perform for him to distinguish the two processes. Note that, even though it is not the main purpose, APTE can also be used to verify reachability properties on a protocol.

Theoretical foundations. APTE relies on symbolic traces, that is a finite representation of infinitely many traces, to decide the equivalence. In particular from each symbolic trace of the two processes is extracted two sets of *constraint systems*. The two processes are then equivalent if and only if all pairs of sets of constant systems are symbolically equivalent. The algorithm used in APTE to decide the symbolic equivalence between sets of constraint systems was proved to be complete, sound and to always terminate in [7,5]. It relies on a set of rules on constraint systems that simplify the sets of constraint systems given as input to render the decision trivial. The extension to length trace equivalence between processes was presented and proved in [8].

Implementation details. The tool is implemented in Ocaml[1] and the source code has about 12Klocs. The source code is highly modular: each mathematical notion used in the algorithm is implemented in a separate module. To facilitate any new extension and optimisation of the tool, the data structures are always hidden in the modules, *i.e.* we only use abstract types (sometimes called opaque types) in the interface files. The format of the comments in the interface files is the one of Ocamldoc that generates a LaTex file with the documented interfaces.

Availability. APTE is an open source software and is distributed under GNU General Public Licence 3.0. It can be downloaded at `http://projects.lsv.ens-cachan.fr/APTE/` where a mailing list, some relevant examples and also a list of publications related or using APTE is available.

[1] `http://caml.inria.fr`

Anonymity on PrA	**Status**	**Execution time**
Original	satisfy trace equivalence but length attack	0.01 sec
Fix (one session)	safe	0.08 sec
Fix (two sessions)	safe	2 hours
Fix (three sessions)	safe	$>$ 2 days

Unlinkability on BAC	**Status**	**Execution time**
French	attack	0.09 sec
UK	safe ?	$>$ 2 days

Unkinkability on PaA	**Status**	**Execution time**
Original	length attack	0.08 sec
Fix	safe	2.8 sec

Others	**Status**	**Execution time**
Anonymity on PaA	safe	3.2 sec
Needham-Shroeder	attack	0.01 sec
Needham-Shroeder-Love	safe	0.4 sec

These results were obtained by using APTE on a 2.9 Ghz Intel Core i7, 8 GB DDR3.

Fig. 1. Experimental results

Upcoming features. Nowadays, all computers are equipped with a multi-core processor and sometimes with several processors, thus we are currently working on a distributed version of APTE that will take advantage of such multi-core processors and clusters. Moreover, we would like to include new cryptographic primitives such as XOR, blind signature and re-encryption used in very interesting protocols *e.g.* Caveat Coercitor [11].

4 Experimental results

We use APTE on several case studies found in the literature. The figure 1 summarises the results. In particular, we focused on the Private Authentication (PrA) protocol [2] and the protocols of the electronic passport (a description of the protocols can be found in [5]). The two key results that we obtained using APTE are a new attack on the anonymity of the Private Authentication protocol and a new attack on the unlinkability of the Passive Authentication protocol (PaA) of the electronic passport. Both attacks rely on the attacker being able to observe the length of messages. In both cases, we propose possible fixes and show their security with APTE for few sessions of the protocols. Observe that execution times for the trace equivalence and length trace equivalence are very similar. However, depending on how many sessions you consider, the execution time varies greatly. For example, in the case of the Private Authentication protocol, one sessions is computed in less than a second whereas two sessions take

a couple of hours and three sessions take more than two days. Using APTE, we also rediscovered an existing attack on the unlinkability of the Basic Access Control protocol (BAC) used in the French electronic passport. Note that proving the unlinkability of the BAC protocol for the UK passport took too much time and so we stopped the execution after two days. We applied APTE to prove the anonymity of the Passive Authentication protocol. At last, since all reachability properties can be expressed by an equivalence, APTE is also able to find the very classical attack on the secrecy of the Needham-Schroeder protocol and prove the secrecy on the Needham-Schroeder-Love protocol.

References

1. Machine readable travel document. Technical Report 9303, International Civil Aviation Organization, 2008.
2. M. Abadi and C. Fournet. Private authentication. *Theoretical Computer Science*, 322(3):427–476, 2004.
3. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th Computer Security Foundations Workshop (CSFW'01)*, 2001.
4. 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.
5. V. Cheval. *Automatic verification of cryptographic protocols: privacy-type properties.* Phd thesis, ENS Cachan, France, 2012.
6. V. Cheval and B. Blanchet. Proving more observational equivalences with proverif. In David Basin and John Mitchell, editors, *Proceedings of the 2nd International Conference on Principles of Security and Trust (POST'13*, Lecture Notes in Computer Science, pages 226–246, Roma, Italy, March 2013. Springer.
7. V. Cheval, H. Comon-Lundh, and S. Delaune. Trace equivalence decision: Negative tests and non-determinism. In *18th ACM Conference on Computer and Communications Security (CCS'11)*, 2011.
8. V. Cheval, V. Cortier, and A. Plet. Lengths may break privacy – or how to check for equivalences with length. In *Proceedings of the 25th International Conference on Computer Aided Verification (CAV'13)*, pages 708–723. Springer, July 2013.
9. Ş. Ciobâcă. *Automated Verification of Security Protocols with Applications to Electronic Voting.* Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, December 2011.
10. Cas J.F. Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 119–128, New York, NY, USA, 2008. ACM.
11. G. Grewal, M. Ryan, S. Bursuc, and P. Ryan. Caveat coercitor: Coercion-evidence in electronic voting. In *IEEE Symposium on Security and Privacy*, pages 367–381, 2013.
12. A. Tiu and J. Dawson. Automating open bisimulation checking for the spi calculus. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 307–321. IEEE Computer Society Press, 2010.
13. L. Vigan. Automated security protocol analysis with the avispa tool. In *Proceedings of the XXI Mathematical Foundations of Programming Semantics (MFPS'05)*, volume 155 of *ENTCS*, pages 61–86. Elsevier, 2006.