D2.3 Implementation of prototypes for equivalence properties

Steve Kremer

January 16, 2012

The results presented in this report have been obtained by Mathieu Baudet, Rohit Chadha, Ştefan Ciobâcă, Vincent Cheval, Hubert Comon-Lundh, Véronique Cortier and Steve Kremer.

Abstract

Equivalence properties are extremely useful in the analysis of security protocols for expressing privacy type properties. While reachability properties have been extensively studies and benefit from rich tool support the situation for equivalence properties is different. In this review we give an overview of four prototype tools that have been developed in the AVOTÉ project.

1 Introduction

Automated analysis of security protocols has proved extremely successful, and there are several automated tools, e.g., [Bla01, A^+05, EMM09], that can automatically check trace-properties such as (weak forms of) confidentiality and authentication. While trace-based properties are certainly important, many crucial security properties can only be expressed in terms of *indistinguishability* (or equivalence). They include strong flavors of confidentiality [Bla04]; resistance to guessing attacks in password based protocols [Bau05]; and anonymity properties in private authentication [AF04], electronic voting [DKR09, BHM08], vehicular networks [DDS10] and RFID protools [ACRR10, BCdH10]. More generally, indistinguishability allows to model security by the means of ideal systems, which are correct by construction [AG99, DKP09]. Indistinguishability properties of cryptographic protocols are naturally modeled by the means of *observational* and *testing equivalences* in cryptographic extensions of process calculi, e.g., the spi [AG99] and the applied-pi calculus [AF01]. While we have good tools for automated verification of trace properties, the situation is different for indistinguishability properties.

Many decidability results have been obtained in the restricted case of a pure eavesdropper, i.e., a passive adversary, for *static equivalence* [AC06, CD07, ACD07]. However, these results do not provide tool support nor practical algorithms ready for implementation. In the case of an active adversary Hüttel [Hüt02] showed undecidability of observational equivalence in the spi calculus, even for the finite control fragment, as well as decidability for the finite, i.e., replication-free, fragment of the spi calculus. The decidability result however only holds for a fixed set of cryptographic primitives and does not yield a practical algorithm. Current results [BAF05] allow to approximate observational equivalence for an unbounded number of sessions. However, this approximation does not suffice to conclude for many applications, *e.g.*, [DKR09, ACRR10]. Symbolic bisimulations have also been devised for the spi [BBN04, Bor08, TD10] and applied pi calculus [DKR10, LL10] to avoid unbounded branching due to adversary inputs. However, only [DKR10, TD10] and [BBN04] yield a decision procedure, again only approximating observational equivalence. The results of [DKR10] have been further refined to show a decision procedure on a restricted class of *simple* processes [CD09]. They rely on a procedure deciding the equivalence of constraint systems, introduced by Baudet [Bau05], for the special case of verifying the existence of guessing attacks. Baudet's procedure allows arbitrary cryptographic primitives that can be modeled as a subterm convergent rewrite systems [AC06]. An alternate procedure achieving the same goal was proposed by Chevalier and Rusinowitch [CR10]. However, both procedures are highly non-deterministic and do not yield a reasonable algorithm that could be implemented. This was one of the motivations for Cheval *et al.* [CCLD10] to designed a new procedure and a prototype tool to decide the equivalence of constraint systems described in this report. Their result only holds for a fixed set of primitives which is insufficient for many examples stemming from electronic voting. Overcoming this was one of the motivations when developing the AKiSs tool also described below. AKiSs is able to decide trace equivalence for a class of determinate processes and to approximate trace equivalence for a more general class. The tool supports a wide range of equational theories, notably all optimally reducing convergent theories (including subterm convergent theories, as well as theories for blind signatures and trapdoor commitment).

In this report we overview four prototypes that gave been implemented during the AVOTÉ project. All tools are freely available for download. We attach the accompanying publications [BCD09, CDK11, CCLD10, CCK12] for each tool.

- YAPA (Yet Another Protocol Analyzer) is a tool dedicated to the analysis of deducibility and static equivalence.

  `http://www.lsv.ens-cachan.fr/~baudet/yapa/`

- KiSs (Knowledge in Security protocolS) is a tool for deciding deduction and static equivalence under certain convergent term rewriting systems.

  `http://www.lsv.ens-cachan.fr/~ciobaca/kiss/`

- ADECS decides symbolic trace equivalence where the inputs are given as deducible constraint systems.

  `http://www.lsv.ens-cachan.fr/~cheval/program/adecs/`

- AKiSs (Active Knowledge in Security protocolS) is a tool for automatically checking trace equivalence for a bounded number of sessions in cryptographic protocols.

  `http://www.lsv.ens-cachan.fr/~ciobaca/akiss/`

2 YAPA and KiSs

YAPA implements a generic decision procedure for static equivalence. YAPA takes as input any convergent rewrite system. We show that the algorithm covers all previously existing decision procedures for convergent theories. We prove the algorithm sound and complete, up to explicit failure cases. Note that (unfailing) termination cannot be guaranteed in general since the problem of checking deducibility and static equivalence is undecidable, even for convergent theories [AC06]. To address this issue and turn our algorithm into a decision procedure for a given convergent theory, we provide two criteria. First, we define a syntactic criterion on the rewrite rules that ensures that the algorithm never fails. This criterion is enjoyed in particular by any convergent subterm theory, as well as the theories of blind signature and homomorphic encryption. Second, we provide a termination criterion based on deducibility: provided that failure cannot occur, termination on a given input is equivalent to the existence of some natural finite representation of deducible terms.

KiSs implements a similar generic procedure. However it uses a more general representation of deducible terms to overcome some limitation of the procedure implemented in YAPA. The procedure terminates on a wide range of equational theories. In particular, we obtain a new decidability result for the theory of trapdoor bit commitment encountered when studying electronic voting protocols which is out of the scope of YAPA. Given the more complex representation of deducible terms termination proofs are significantly more complicated. While termination has been shown for subterm convergent theories, and theories for blind signatures and trapdoor bit commitments, it is still an open question whether the tool terminates on all theories on which YAPA terminates.

3 ADECS

In ADECS an infinite sets of possible traces are symbolically represented using deducibility constraints. We give a new algorithm that decides the trace equivalence for the traces that are represented using such constraints, in the case of signatures, symmetric and asymmetric encryptions. The main idea of our method is to simultaneously solve pairs of constraints, instead of solving each constraint separately and comparing the solutions, as in [Bau05]. These pairs are successively split into several pairs of systems, while pre- serving the symbolic equivalence: roughly, the father pair is in the relation if, and only if, all the sons pairs are in the relation. This is not fully correct, since, for termination purposes, we need to keep track of some earlier splitting, using ad- ditional predicates. Such predicates, together with the constraint systems, yield another notion of equivalence, which is preserved upwards, while the former is preserved downwards. When a pair of constraints cannot be split any more, then the equivalence can be trivially checked. This is the first implemented algorithm, deciding symbolic trace equivalence.

4 AKiSs

In AKiSs we introduce a new procedure for verifying equivalence properties for processes specified in a cryptographic process calculus (without replication). The main contributions are as follows.

- Our procedure automatically checks for two equivalences $\approx_{ct}$ and $\approx_{ft}$ which over- and under-approximate the standard notion of trace equivalence $\approx_t$ for cryptographic protocols: $\approx_{ft}$ can be used to prove protocols correct while $\approx_{ct}$ can be used to rule out incorrect protocols.

- Cortier and Delaune [CD09] have shown that observational equivalence coincides with $\approx_t$ for the class of *determinate* processes. They also give a decision procedure for a strict sub-class of determinate processes, namely, *simple* processes. We show that for determinate processes the coarser relation $\approx_{ct}$ coincides with $\approx_t$, and our procedure can be used to verify observational equivalence for the whole class of determinate processes.

- A novelty of our procedure is that it is based on a *fully abstract* modeling of symbolic traces in *first-order Horn clauses*. This is in contrast to the constraint-solving techniques employed in [TD10, CCLD10, CCLD11, Bau05, CR10] for verifying under-approximations of observational equivalence. Techniques based on Horn clauses have been extensively used, e.g., in [Bla01, Wei99, Gou05], for an unbounded number of sessions. Of these tools, only ProVerif [Bla01, BAF05] can verify an equivalence property, which is an under-approximation of observational equivalence. Horn clause modeling of an unbounded number of sessions of security protocols may allow false attacks. In contrast, we show our modeling of a bounded number of sessions for determinate protocols to be precise.

- Our modeling is fully abstract for arbitrary cryptographic primitives that can be modeled as a convergent rewrite system which has the *finite variant property*. Not only this strictly includes the class of primitives that can be modeled as subterm convergent rewrite systems, but this also allows us to handle a larger class of cryptographic primitives than [TD10, CCLD10, CCLD11, Bau05, CR10, Bla01]. For example, this allows us to handle trapdoor commitment as used by Okamoto for electronic voting in [Oka97]. Although we were unable to prove termination of our procedure, we conjecture it to terminate for the class of cryptographic primitives that can be modeled as subterm convergent rewrite systems. Our conjecture is supported by experimental evidence.

- The AKiSs tool was used among others to give the first automated proof of anonymity for the electronic voting protocols presented in [FOO92] and [Oka97].

References

[A$^+$05] Alessandro Armando et al. The AVISPA tool for the automated validation of internet security protocols and applications. In *17th International Conference on Computer Aided Verification (CAV'05)*, LNCS, pages 281–285. Springer, 2005.

[AC06] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, 2006.

[ACD07] Mathilde Arnaud, Véronique Cortier, and Stéphanie Delaune. Combining algorithms for deciding knowledge in security protocols. In *International Symposium on Frontiers of Combining Systems (FroCoS'07)*, volume 4720 of *LNAI*, pages 103–117. Springer, 2007.

[ACRR10] Myrto Arapinis, Tom Chothia, Eike Ritter, and Mark D. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Comp. Soc. Press, 2010.

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

[AF04] Martín Abadi and Cédric Fournet. Private authentication. *Theoretical Computer Science*, 322(3):427–476, 2004.

[AG99] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. Comput.*, 148(1):1–70, 1999.

[BAF05] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *Symposium on Logic in Computer Science*, pages 331–340. IEEE Comp. Soc. Press, 2005.

[Bau05] Mathieu Baudet. Deciding security of protocols against off-line guessing attacks. In *12th Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM Press, 2005.

[BBN04] Johannes Borgström, Sébastien Briais, and Uwe Nestmann. Symbolic bisimulation in the spi calculus. In *15th Int. Conference on Concurrency Theory (CONCUR'04)*, volume 3170 of *LNCS*, pages 161–176. Springer, 2004.

[BCD09] Mathieu Baudet, Véronique Cortier, and Stéphanie Delaune. YAPA: A generic tool for computing intruder knowledge. In *20th International Conference on Rewriting Techniques and Applications (RTA'09)*, volume 5595 of *LNCS*, pages 148–163. Springer, 2009.

[BCdH10] Mayla Bruso, Konstantinos Chatzikokolakis, and Jerry den Hartog. Analysing unlinkability and anonymity using the applied pi calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Comp. Soc. Press, 2010.

[BHM08] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *21st Computer Security Foundations Symposium (CSF'08)*. IEEE Comp. Soc. Press, 2008.

[Bla01] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Comp. Soc. Press, 2001.

[Bla04] Bruno Blanchet. Automatic proof of strong secrecy for security protocols. In *Symposium on Security and Privacy (S&P'04)*, pages 86–100, 2004.

[Bor08] Johannes Borgström. *Equivalences and Calculi for Formal Verifiation of Cryptographic Protocols*. Phd thesis, EPFL, Switzerland, 2008.

[CCK12] Rohit Chadha, Ştefan Ciobâcă, and Steve Kremer. Automated verification of equivalence properties of cryptographic protocols. In Helmut Seidl, editor, *Programming Languages and Systems — Proceedings of the 22nd European Symposium* *on Programming (ESOP'12)*, Lecture Notes in Computer Science, Tallinn, Estonia, March 2012. Springer. To appear.

[CCLD10] Vincent Cheval, Hubert Comon-Lundh, and Stephanie Delaune. Automating security analysis: symbolic equivalence of constraint systems. In *International Joint Conference on Automated Reasoning (IJCAR'10)*, LNAI, pages 412–426. Springer, 2010.

[CCLD11] Vincent Cheval, Hubert Comon-Lundh, and Stéphanie Delaune. Trace equivalence decision: Negative tests and non-determinism. In *18th Conference on Computer and Communications Security (CCS'11)*, pages 321–330. ACM Press, 2011.

[CD07] Véronique Cortier and Stéphanie Delaune. Deciding knowledge in security protocols for monoidal equational theories. In *14th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, volume 4790 of *LNAI*, pages 196–210. Springer, 2007.

[CD09] Véronique Cortier and Stéphanie Delaune. A method for proving observational equivalence. In *22nd Computer Security Foundations Symposium (CSF'09)*, pages 266–276. IEEE Comp. Soc. Press, 2009.

[CDK11] Ştefan Ciobâcă, Stéphanie Delaune, and Steve Kremer. Computing knowledge in security protocols under convergent equational theories. *Journal of Automated Reasoning*, 2011. To appear.

[CR10] Yannick Chevalier and Michaël Rusinowitch. Decidability of equivalence of symbolic derivations. *Journal of Automated Reasoning*, 2010. To appear.

[DDS10] Morten Dahl, Stéphanie Delaune, and Graham Steel. Formal analysis of privacy for vehicular mix-zones. In *15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *LNCS*, pages 55–70. Springer, 2010.

[DKP09] Stéphanie Delaune, Steve Kremer, and Olivier Pereira. Simulation based security in the applied pi calculus. In *29th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'09)*, volume 4 of *Leibniz International Proceedings in Informatics*, pages 169–180. Leibniz-Zentrum für Informatik, 2009.

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

[DKR10] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Symbolic bisimulation for the applied pi calculus. *Journal of Computer Security*, 18(2):317–377, March 2010.

[EMM09] Santiago Escobar, Catherine Meadows, and José Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design V*, volume 5705 of *LNCS*, pages 1–50. Springer, 2009.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazui Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology — AUSCRYPT '92*, volume 718 of *LNCS*, pages 244–251. Springer, 1992.

[Gou05] Jean Goubault-Larrecq. Deciding $\mathcal{H}_1$ by resolution. *Information Processing Letters*, 95(3):401–408, 2005.

[Hüt02] Hans Hüttel. Deciding framed bisimilarity. In *4th International Workshop on Verification of Infinite-State Systems (INFINITY'02)*, pages 1–20, 2002.

[LL10] Jia Liu and Huimin Lin. A complete symbolic bisimulation for full applied pi calculus. In *36th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'10)*, volume 5901 of *LNCS*, pages 552–563. Springer, 2010.

[Oka97] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *5th Int. Security Protocols Workshop*, volume 1361 of *LNCS*, pages 25–35. Springer, 1997.

[TD10] Alwen Tiu and Jeremy Dawson. Automating open bisimulation checking for the spi-calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 307–321. IEEE Comp. Soc. Press, 2010.

[Wei99] Christoph Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *16th International Conference on Automated Deduction (CADE'99)*, volume 1632 of *LNCS*, pages 314–328. Springer, 1999.

YAPA: A generic tool for computing intruder knowledge [⋆]

Mathieu Baudet[1], Véronique Cortier[2], and Stéphanie Delaune[3]

[1] DCSSI, France
[2] LORIA, CNRS & INRIA project Cassis, France
[3] LSV, ENS Cachan & CNRS & INRIA, France

Abstract. Reasoning about the knowledge of an attacker is a necessary step in many formal analyses of security protocols. In the framework of the applied pi calculus, as in similar languages based on equational logics, knowledge is typically expressed by two relations: deducibility and static equivalence. Several decision procedures have been proposed for these relations under a variety of equational theories. However, each theory has its particular algorithm, and none has been implemented so far.

We provide a generic procedure for deducibility and static equivalence that takes as input any convergent rewrite system. We show that our algorithm covers all the existing decision procedures for convergent theories. We also provide an efficient implementation, and compare it briefly with the more general tool ProVerif.

1 Introduction

Understanding security protocols often requires reasoning about the information accessible to an online attacker. Accordingly, many formal approaches to security rely on a notion of *deducibility* [18, 19] that models whether a piece of data, typically a secret, is retrievable from a finite set of messages. Deducibility, however, does not always suffice to reflect the knowledge of an attacker. Consider for instance a protocol sending an encrypted Boolean value, say, a vote in an electronic voting protocol. Rather than deducibility, the key idea to express confidentiality of the plaintext is that an attacker should not be able to *distinguish* between the sequences of messages corresponding to each possible value.

In the framework of the applied pi-calculus [3], as in similar languages based on equational logics [10], indistinguishability corresponds to a relation called *static equivalence*: roughly, two sequences of messages are *statically equivalent* when they satisfy the same algebraic relations from the attacker's point of view. Static equivalence plays an important role in the study of guessing attacks (e.g. [13, 5, 1]), as well as for anonymity properties and electronic voting protocols (e.g. [17]). In several cases, this notion has also been shown to imply the more complex and precise notion of cryptographic indistinguishability [8, 1], related to probabilistic polynomial-time Turing machines.

⋆ Kindly supported by ANR-07-SESU-002 AVOTÉ and ARA SSIA FormaCrypt.

We emphasize that both deducibility and static equivalence apply to observations on finite sets of messages, and do not take into account the dynamic behavior of protocols. Nevertheless, deducibility is used as a subroutine by many general decision procedures [12, 11]. Besides, it has been shown that observational equivalence in the applied pi-calculus coincides with labeled bisimulation [3], that is, corresponds to checking a number of static equivalences and some standard bisimulation conditions.

Deducibility and static equivalence rely on an underlying equational theory for axiomatizing the properties of cryptographic functions. Many decision procedures [2, 14] have been proposed to compute these relations under a variety of equational theories, including symmetric and asymmetric encryptions, signatures, exclusive OR, and homomorphic operators. However, except for the class of subterm convergent theories [2], which covers the standard flavors of encryption and signature, each of these decision results introduces a new procedure, devoted to a particular theory. Even in the case of the general decidability criterion given in [2], we note that the algorithm underlying the proof has to be adapted for each theory, depending on how the criterion is fulfilled.

Perhaps as a consequence of this fact, none of these decision procedures has been implemented so far. Up to our knowledge the only tool able to verify static equivalence is ProVerif [9, 10]. This general tool can handle various equational theories and analyze security protocols under active adversaries. However termination of the verifier is not guaranteed in general, and protocols are subject to (safe) approximations.

The present work aims to fill this gap between theory and implementation and propose an efficient tool for deciding deducibility and static equivalence in a uniform way. It is initially inspired from a procedure for solving more general constraint systems related to active adversaries and equivalence of finite processes, presented in [5], with corrected extended version in [6] (in French). However, due to the complexity of the constraint systems, this decision procedure was only studied for subterm convergent theories, and remains too complex to enable an efficient implementation.

Our first contribution is to provide and study a generic procedure for checking deducibility and static equivalence, taking as input any convergent theory (that is, any equational theory described by a finite convergent rewrite system). We prove the algorithm sound and complete, up to explicit failure cases. Note that (unfailing) termination cannot be guaranteed in general since the problem of checking deducibility and static equivalence is undecidable, even for convergent theories [2]. To address this issue and turn our algorithm into a decision procedure for a given convergent theory, we provide two criteria. First, we define a syntactic criterion on the rewrite rules that ensures that the algorithm never fails. This criterion is enjoyed in particular by any convergent subterm theory, as well as the theories of blind signature and homomorphic encryption. Termination often follows from a simple analysis of the rules of the algorithm: as a proof of concept, we obtain a new decidability result for deducibility and static equivalence for the prefix theory, representing encryption in CBC mode.

Second, we provide a termination criterion based on deducibility: provided that failure cannot occur, termination on a given input is equivalent to the existence of some natural finite representation of deducible terms. As a consequence, we obtain that our algorithm can decide deducibility and static equivalence for all the convergent theories previously known to be decidable [2].

Our second contribution is an efficient implementation of this generic procedure, called YAPA. After describing the main features of the implementation, we report several experiments suggesting that our tool computes static equivalence faster and for more convergent theories than the general tool ProVerif [9, 10]. Because of space constraints, proofs are in an extended version of this paper [7].

2 Preliminaries

2.1 Term algebra

We start by introducing the necessary notions to describe cryptographic messages in a symbolical way. For modeling cryptographic primitives, we assume a given set of *function symbols* $\mathcal{F}$ together with an arity function $\mathrm{ar} : \mathcal{F} \to \mathbb{N}$. Symbols in $\mathcal{F}$ of arity 0 are called *constants*. We consider a set of *variables* $\mathcal{X}$ and a set of additional constants $\mathcal{W}$ called *parameters*. The (usual, first-order) term algebra generated by $\mathcal{F}$ over $\mathcal{W}$ and $\mathcal{X}$ is written $\mathcal{F}[\mathcal{W} \cup \mathcal{X}]$ with elements denoted by $T, U, T_1 \ldots$ More generally, we write $\mathcal{F}'[A]$ for the least set of terms containing a set A and stable by application of symbols in $\mathcal{F}' \subseteq \mathcal{F}$.

We write $\mathrm{var}(T)$ (resp. $\mathrm{par}(T)$) for the set of variables (resp. parameters) that occur in a term T. These notations are extended to tuples and sets of terms in the usual way. The set of positions (resp. subterms) of a term T is written $\mathrm{pos}(T) \subseteq \mathbb{N}^*$ (resp. $\mathrm{st}(T)$). The subterm of T at position $p \in \mathrm{pos}(T)$ is written $T|_p$. The term obtained by replacing $T|_p$ with a term U in T is denoted $T[U]_p$.

A *(finite, partial) substitution* σ is a mapping from a finite subset of variables, called its *domain* and written $\mathrm{dom}(\sigma)$, to terms. The *image* of a substitution is its image as a mapping $\mathrm{im}(\sigma) = \{\sigma(x) \mid x \in \mathrm{dom}(\sigma)\}$. Substitutions are extended to endomorphisms of $\mathcal{F}[\mathcal{X} \cup \mathcal{W}]$ as usual. We use a postfix notation for their application. A term T (resp. a substitution σ) is *ground* iff $\mathrm{var}(T) = \emptyset$ (resp. $\mathrm{var}(\mathrm{im}(\sigma)) = \emptyset$).

For our cryptographic purposes, it is useful to distinguish a subset $\mathcal{F}_{\mathsf{pub}}$ of $\mathcal{F}$, made of *public function symbols*, that is, intuitively, the symbols made available to the attacker. A *recipe* (or *second-order term*) M, N, $M_1\ldots$ is a term in $\mathcal{F}_{\mathsf{pub}}[\mathcal{W} \cup \mathcal{X}]$, that is, a term containing no *private* (non-public) function symbols. A *plain term* (or *first-order term*) t, r, s, $t_1\ldots$ is a term in $\mathcal{F}[\mathcal{X}]$, that is, containing no parameters. A *(public, ground, non-necessarily linear) n-ary context* C is a recipe in $\mathcal{F}_{\mathsf{pub}}[\mathsf{w}_1, \ldots, \mathsf{w}_n]$, where we assume a fixed countable subset of parameters $\{\mathsf{w}_1, \ldots, \mathsf{w}_n, \ldots\} \subseteq \mathcal{W}$. If C is a n-ary context, $C[T_1, \ldots, T_n]$ denotes the term obtained by replacing each occurrence of w_i with T_i in C.

2.2 Rewriting

A *rewrite system* $\mathcal{R}$ is a finite set of *rewrite rules* $l \to r$ where $l, r \in \mathcal{F}[\mathcal{X}]$ and $\mathrm{var}(r) \subseteq \mathrm{var}(l)$. A term S *rewrites* to T by $\mathcal{R}$, denoted $S \to_{\mathcal{R}} T$, if there exist

$l \to r$ in $\mathcal{R}$, $p \in \mathrm{pos}(S)$ and a substitution σ such that $S|_p = l\sigma$ and $T = S[r\sigma]_p$. We write $\to^+_{\mathcal{R}}$ for the transitive closure of $\to_{\mathcal{R}}$, $\to^*_{\mathcal{R}}$ for its reflexive and transitive closure, and $=_{\mathcal{R}}$ for its reflexive, symmetric and transitive closure.

A rewrite system $\mathcal{R}$ is *convergent* if is *terminating*, i.e. there is no infinite chains $T_1 \to_{\mathcal{R}} T_2 \to_{\mathcal{R}} \ldots$, and *confluent*, i.e. for every terms S, T such that $S =_{\mathcal{R}} T$, there exists U such that $S \to^*_{\mathcal{R}} U$ and $T \to^*_{\mathcal{R}} U$.

A term T is *$\mathcal{R}$-reduced* if there is no term S such that $T \to_{\mathcal{R}} S$. If $T \to^*_{\mathcal{R}} S$ and S is $\mathcal{R}$-reduced then S is *a $\mathcal{R}$-reduced form of* T. When this reduced form is unique (in particular if $\mathcal{R}$ is convergent), we write $S = T{\downarrow_{\mathcal{R}}}$.

2.3 Equational theories

We equip the signature $\mathcal{F}$ with an equational theory represented by a set of equations $\mathcal{E}$ of the form $s = t$ with $s, t \in \mathcal{F}[\mathcal{X}]$. The equational theory E generated by $\mathcal{E}$ is the least set of equations containing $\mathcal{E}$ that is stable under the axioms of congruence (reflexivity, symmetry, transitivity, application of function symbols) and under application of substitutions. We write $=_{\mathsf{E}}$ for the corresponding relation on terms. Equational theories have proved very useful for modeling algebraic properties of cryptographic primitives [15, 2].

We are particularly interested in theories E that can be represented by a convergent rewrite system $\mathcal{R}$, i.e. theories for which there exists a convergent rewrite system $\mathcal{R}$ such that the two relations $=_{\mathcal{R}}$ and $=_{\mathsf{E}}$ coincide. The rewrite system $\mathcal{R}$ —and by extension the equational theory E— is *subterm convergent* if, in addition, we have that for every rule $l \to r \in \mathcal{R}$, r is either a subterm of l or a ground $\mathcal{R}$-reduced term. This class encompasses the one of the same name used in [2], the class of dwindling theories used in [4], and the class of public-collapsing theories introduced in [16].

Example 1. Consider the signature $\mathcal{F}_{\mathsf{enc}} = \{\mathsf{dec}, \mathsf{enc}, \langle \_, \_ \rangle, \pi_1, \pi_2\}$. The symbols dec, enc and $\langle \_, \_ \rangle$ are functional symbols of arity 2 that represent respectively the decryption, encryption and pairing functions, whereas π_1 and π_2 are functional symbols of arity 1 that represent the projection function on the first and the second component of a pair, respectively. The equational theory of pairing and symmetric (deterministic) encryption, denoted by $\mathsf{E}_{\mathsf{enc}}$, is generated by the equations $\mathcal{E}_{\mathsf{enc}} = \{\mathsf{dec}(\mathsf{enc}(x, y), y) = x,\ \ \pi_1(\langle x, y\rangle) = x,\ \ \pi_2(\langle x, y\rangle) = y\}$.

Motivated by the modeling of the ECB mode of encryption, we may also consider an encryption symbol that is homomorphic with respect to pairing:

$$\mathcal{E}_{\mathsf{hom}} = \mathcal{E}_{\mathsf{enc}} \cup \left\{ \begin{array}{l} \mathsf{enc}(\langle x, y\rangle, z) = \langle \mathsf{enc}(x, z), \mathsf{enc}(y, z)\rangle \\ \mathsf{dec}(\langle x, y\rangle, z) = \langle \mathsf{dec}(x, z), \mathsf{dec}(y, z)\rangle \end{array} \right\}.$$

If we orient the equations from left to right, we obtain two rewrite systems $\mathcal{R}_{\mathsf{enc}}$ and $\mathcal{R}_{\mathsf{hom}}$. Both rewrite systems are convergent, only $\mathcal{R}_{\mathsf{enc}}$ is subterm convergent.

From now on, we assume a given equational theory E represented by a convergent rewrite system $\mathcal{R}$. A symbol f is *free* if f does not occur in $\mathcal{R}$. In order to model (an unbounded number of) random values possibly generated by the attacker, we assume that $\mathcal{F}_{\mathsf{pub}}$ contains infinitely many free public constants. We will use free private constants to model secrets, for instance the secret keys used to encrypt a message. Private (resp. public) free constants are closely related to bound (resp. free) *names* in the framework of the applied pi calculus [3]. Our formalism also allows one to consider non-constant private symbols.

3 Deducibility and static equivalence

In order to describe the cryptographic messages observed or inferred by an attacker, we introduce the following notions of deduction facts and frames.

A *deduction fact* is a pair, written $M \rhd t$, made of a recipe $M \in \mathcal{F}_{\mathsf{pub}}[\mathcal{W} \cup \mathcal{X}]$ and a plain term $t \in \mathcal{F}[\mathcal{X}]$. Such a deduction fact is *ground* if $\mathrm{var}(M, t) = \emptyset$. A *frame*, denoted by letters φ, Φ, $\Phi_0 \ldots$, is a finite set of ground deduction facts. The *image* of a frame is defined by $\mathrm{im}(\Phi) = \{t \mid M \rhd t \in \Phi\}$. A frame Φ is *one-to-one* if $M_1 \rhd t$, $M_2 \rhd t \in \Phi$ implies $M_1 = M_2$.

A frame φ is *initial* if it is of the form $\varphi = \{w_1 \rhd t_1, \ldots, w_\ell \rhd t_\ell\}$ for some distinct parameters $w_1, \ldots, w_\ell \in \mathcal{W}$. Initial frames are closely related to the notion of frames in the applied pi-calculus [3]. The parameters w_i can be seen as labels that refer to the messages observed by an attacker. Given such an initial frame φ, we denote by $\mathrm{dom}(\varphi)$ its *domain* $\mathrm{dom}(\varphi) = \{w_1, \ldots, w_\ell\}$. If $\mathrm{par}(M) \subseteq \mathrm{dom}(\varphi)$, we write $M\varphi$ for the term obtained by replacing each w_i by t_i in M. If in addition M is ground then $t = M\varphi$ is a ground plain term.

3.1 Deducibility, recipes

Classically (see e.g. [2]), a ground term t is *deducible* modulo E from an initial frame φ if there exists $M \in \mathcal{F}_{\mathsf{pub}}[\mathrm{dom}(\varphi)]$ such that $M\varphi =_{\mathsf{E}} t$. This corresponds to the intuition that the attacker may compute (infer) t from φ. For the purpose of our study, we generalize this notion to arbitrary frames, and even sets of (non-necessarily ground) deduction facts ϕ, using the notations $\rhd_\phi$ and $\rhd_\phi^{\mathsf{E}}$.

Definition 1 (deducibility). *Let ϕ be finite set of deductions facts, for instance a frame. We say that M* is a recipe of t in ϕ, *written $M \rhd_\phi t$, iff there exist a (public, ground, non-necessarily linear) n-ary context C and some deduction facts $M_1 \rhd t_1, \ldots, M_n \rhd t_n$ in ϕ such that $M = C[M_1, \ldots, M_n]$ and $t = C[t_1, \ldots, t_n]$. In that case, we say that t is* syntactically deducible *from ϕ, also written $\phi \vdash t$.*

We say that M is a recipe of t in ϕ modulo E, *written $M \rhd_\phi^{\mathsf{E}} t$, iff there exists a term t' such that $M \rhd_\phi t'$ and $t' =_{\mathsf{E}} t$. In that case, we say that t is* deducible from ϕ modulo E, *written $\phi \vdash_{\mathsf{E}} t$.*

We note that $M \rhd_\varphi t$ is equivalent to $M\varphi = t$ when φ is an initial frame and when t (or equivalently M) is ground.

Example 2. Consider the equational theory $\mathsf{E}_{\mathsf{enc}}$ given in Example 1. Let $\varphi = \{\mathsf{w}_1 \rhd \langle \mathsf{enc}(\mathsf{s}_1, \mathsf{k}), \mathsf{enc}(\mathsf{s}_2, \mathsf{k})\rangle, \mathsf{w}_2 \rhd \mathsf{k}\}$ where s_1, s_2 and k are private constant symbols. We have that $\langle \mathsf{w}_2, \mathsf{w}_2\rangle \rhd_\varphi \langle \mathsf{k}, \mathsf{k}\rangle$, and $\mathsf{dec}(\mathsf{proj}_1(\mathsf{w}_1), \mathsf{w}_2) \rhd_\varphi^{\mathsf{E}_{\mathsf{enc}}} \mathsf{s}_1$.

3.2 Static equivalence, visible equations

Deducibility does not always suffice for expressing the knowledge of an attacker. In particular, it does not account for the partial information that an attacker may obtain about secrets. This issue motivates the study of visible equations and static equivalence [3], defined as follows.

Definition 2 (static equivalence). *Let φ be an initial frame. The set of* visible equations of φ modulo E *is defined as*

$$\mathrm{eq}_{\mathsf{E}}(\varphi) = \{M \bowtie N \mid M, N \in \mathcal{F}_{\mathsf{pub}}[\mathrm{dom}(\varphi)],\ M\varphi =_{\mathsf{E}} N\varphi\}$$

where $\bowtie$ is a dedicated commutative symbol. Two initial frames φ_1 and φ_2 with the same domain are statically equivalent *modulo E, written $\varphi_1 \approx_{\mathsf{E}} \varphi_2$, if their sets of visible equations are equal, i.e.* $\mathrm{eq}_{\mathsf{E}}(\varphi_1) = \mathrm{eq}_{\mathsf{E}}(\varphi_2)$.

This definition is in line with static equivalence in the applied pi calculus [3]. For the purpose of finitely describing the set of visible equations $\mathrm{eq}_{\mathsf{E}}(\varphi)$ of an initial frame, we introduce *quantified equations* of the form $\forall z_1, \ldots, z_q.M \bowtie N$ where $z_1, \ldots, z_q \in \mathcal{X}$, $q \geq 0$ and $\mathrm{var}(M, N) \subseteq \{z_1, \ldots, z_q\}$. In the following, finite sets of quantified equations are denoted Ψ, Ψ_0,... We write $\Psi \models M \bowtie N$ when the ground equation $M \bowtie N$ is a consequence of Ψ in the usual, first-order logics with equality axioms for the relation $\bowtie$ (that is, reflexivity, symmetry, transitivity and compatibility with symbols in $\mathcal{F}_{\mathsf{pub}}$). When no confusion arises, we may refer to quantified equations simply as *equations*. As usual, quantified equations are considered up to renaming of bound variables.

Example 3. Consider again the equational theory $\mathsf{E}_{\mathsf{enc}}$ given in Example 1. Let $\varphi_1 = \{\mathsf{w}_1 \rhd \mathsf{enc}(\mathsf{c}_0, \mathsf{k}),\ \mathsf{w}_2 \rhd \mathsf{k}\}$ and $\varphi_2 = \{\mathsf{w}_1 \rhd \mathsf{enc}(\mathsf{c}_1, \mathsf{k}),\ \mathsf{w}_2 \rhd \mathsf{k}\}$ where c_0, c_1 are public constants and k is a private constant. Let $\Psi_1 = \{\mathsf{enc}(\mathsf{c}_0, \mathsf{w}_2) \bowtie \mathsf{w}_1\}$ and $\Psi_2 = \{\mathsf{enc}(\mathsf{c}_1, \mathsf{w}_2) \bowtie \mathsf{w}_1\}$. We have that $\Psi_i \models \mathrm{eq}_{\mathsf{E}_{\mathsf{enc}}}(\varphi_i)$ for $i = 1, 2$. Hence, $\mathrm{eq}_{\mathsf{E}_{\mathsf{enc}}}(\varphi_1) \neq \mathrm{eq}_{\mathsf{E}_{\mathsf{enc}}}(\varphi_2)$ and the two frames φ_1 and φ_2 are not statically equivalent. However, it can be shown that $\{\mathsf{w}_1 \rhd \mathsf{enc}(\mathsf{c}_0, \mathsf{k})\} \approx_{\mathsf{E}_{\mathsf{enc}}} \{\mathsf{w}_1 \rhd \mathsf{enc}(\mathsf{c}_1, \mathsf{k})\}$.

4 Main procedure

In this section, we describe our algorithms for checking deducibility and static equivalence on convergent rewrite systems. After some additional notations, we present the core of the procedure, which consists of a set of transformation rules used to saturate a frame and a finite set of quantified equations. We then show how to use this procedure to decide deducibility and static equivalence, provided that saturation succeeds.

Soundness and completeness of the saturation procedure are detailed in Section 5. We provide sufficient conditions on the rewrite systems to ensure success of saturation in Section 6.

4.1 Decompositions of rewrite rules

Before stating the procedure, we introduce the following notion of *decomposition* to account for the possible superpositions of an attacker's context with a left-hand side of rewrite rule.

Definition 3 (decomposition). *Let n, p, q be non-negative integers. A (n, p, q)-*decomposition *of a term l (and by an extension of any rewrite rule $l \to r$) is a (public, ground, non-necessarily linear) context $D \in \mathcal{F}_{\mathsf{pub}}[\mathcal{W}]$ such that* $\mathrm{par}(D) = \{\mathsf{w}_1, \dots, \mathsf{w}_{n+p+q}\}$ *and $l = D[l_1, \dots, l_n, y_1, \dots, y_p, z_1, \dots, z_q]$ where*

- *$l_1, \dots, l_n$ are mutually-distinct non-variable terms,*
- *$y_1, \dots, y_p$ and $z_1, \dots, z_q$ are mutually-distinct variables, and*
- *$y_1, \dots, y_p \in \mathrm{var}(l_1, \dots, l_n)$ whereas $z_1, \dots, z_q \notin \mathrm{var}(l_1, \dots, l_n)$.*

A decomposition D is proper *if it is not a parameter (i.e. $D \neq \mathsf{w}_1$).*

Example 4. Consider the rewrite rule $\mathsf{dec}(\mathsf{enc}(x, y), y) \to x$. This rule admits two proper decompositions up to permutation of parameters:

- $D_1 = \mathsf{dec}(\mathsf{enc}(\mathsf{w}_1, \mathsf{w}_2), \mathsf{w}_2)$ where $n = 0$, $p = 0$, $q = 2$, $z_1 = x$, $z_2 = y$;
- $D_2 = \mathsf{dec}(\mathsf{w}_1, \mathsf{w}_2)$ where $n = 1$, $p = 1$, $q = 0$, $l_1 = \mathsf{enc}(x, y)$ and $y_1 = y$.

4.2 Transformation rules

To check deducibility and static equivalence, we proceed by saturating an initial frame, adding some deduction facts and equations satisfied by the frame. We consider *states* that are either the failure state $\perp$ or a couple (Φ, Ψ) formed by a one-to-one frame Φ in $\mathcal{R}$-reduced form and a finite set of quantified equations Ψ.

Given an initial frame φ, our procedure starts from an initial state associated to φ, denoted by $\mathrm{Init}(\varphi)$, obtained by reducing φ and replacing duplicated terms by equations. Formally, $\mathrm{Init}(\varphi)$ is the result of a procedure recursively defined as follows: $\mathrm{Init}(\emptyset) = (\emptyset, \emptyset)$, and assuming $\mathrm{Init}(\varphi) = (\Phi, \Psi)$, we have

$$\mathrm{Init}(\varphi \uplus \{w \rhd t\}) = \begin{cases} (\Phi, \Psi \cup \{w \bowtie w'\}) & \text{if there exists some } w' \rhd t\downarrow_{\mathcal{R}} \in \Phi \\ (\Phi \cup \{w \rhd t\downarrow_{\mathcal{R}}\}, \Psi) & \text{otherwise.} \end{cases}$$

The main part of our procedure consists in saturating a state (Φ, Ψ) by means of the transformation rules described in Figure 1. The **A** rules are designed for applying a rewrite step on top of existing deduction facts. If the resulting term is already syntactically deducible then a corresponding equation is added (rule **A.1**); or else if it is ground, the corresponding deduction fact is added to the state (rule **A.2**); otherwise, the procedure may fail (rule **A.3**). The **B** rules are meant to add syntactically deducible subterms (rule **B.2**) or related equations (rule **B.1**). For technical reasons, rule **A.1** is parametrized by a function Ctx with values of the form M or $\perp$, and satisfying the following properties:

(a) if $\phi \vdash t\downarrow_{\mathcal{R}}$, then for any Ψ and α, $\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t, \Psi, \alpha) \neq \perp$;

A. Inferring deduction facts and equations by context reduction
Assume that

$l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$ is a proper decomposition of $(l \to r) \in \mathcal{R}$
$M_1 \rhd t_1, \ldots, M_{n+p} \rhd t_{n+p} \in \Phi$
$(l_1, \ldots, l_n, y_1, \ldots, y_p)\,\sigma = (t_1, \ldots, t_{n+p})$

1. If there exists $M = \mathrm{Ctx}(\Phi \cup \{z_1 \rhd z_1, \ldots, z_q \rhd z_q\} \vdash^?_{\mathcal{R}} r\sigma,\ \Psi,\ (l, r, D, \sigma))$, then
$$(\Phi, \Psi) \Longrightarrow (\Phi, \Psi \cup \{\forall z_1, \ldots, z_q.D[M_1, \ldots, M_{n+p}, z_1 \ldots, z_q] \bowtie M\}) \qquad (\mathbf{A.1})$$
2. Else, if $(r\sigma)\downarrow_{\mathcal{R}}$ is ground, then
$$\begin{aligned}(\Phi, \Psi) \Longrightarrow (&\Phi \cup \{M_0 \rhd (r\sigma)\downarrow_{\mathcal{R}}\},\\ &\Psi \cup \{\forall z_1, \ldots, z_q.D[M_1, \ldots, M_{n+p}, z_1 \ldots, z_q] \bowtie M_0\})\end{aligned} \qquad (\mathbf{A.2})$$
where $M_0 = D[M_1, \ldots, M_{n+p}, \mathsf{a}, \ldots, \mathsf{a}]$ for some fixed public constant a.
3. Otherwise, $(\Phi, \Psi) \Longrightarrow \bot$ $\qquad (\mathbf{A.3})$

B. Inferring deduction facts and equations syntactically
Assume that $M_0 \rhd t_0, \ldots, M_n \rhd t_n \in \Phi \qquad t = f(t_1, \ldots, t_n) \in \mathrm{st}(t_0) \qquad f \in \mathcal{F}_{\mathsf{pub}}$

1. If there exists M such that $(M \rhd t) \in \Phi$,
$$(\Phi, \Psi) \Longrightarrow (\Phi, \Psi \cup \{f(M_1, \ldots, M_n) \bowtie M\}) \qquad (\mathbf{B.1})$$
2. Otherwise, $(\Phi, \Psi) \Longrightarrow (\Phi \cup \{f(M_1, \ldots, M_n) \rhd t\}, \Psi)$ $\qquad (\mathbf{B.2})$

Fig. 1. Transformation rules

(b) if $M = \mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t,\ \Psi, \alpha)$ then there exist M' and s such that $\Psi \models M \bowtie M'$, $M' \rhd_\phi s$ and $t \to^*_{\mathcal{R}} s$. (This justifies the notation $\phi \vdash^?_{\mathcal{R}} t$ used to denote a specific deducibility problem.)

Note that a simple choice for $\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t, \Psi, \alpha)$ is to solve the deducibility problem $\phi \vdash^? t\downarrow_{\mathcal{R}}$ in the empty equational theory, and then return a corresponding recipe M, if any. (This problem is easily solved by induction on $t\downarrow_{\mathcal{R}}$.) Yet, optimizing the function Ctx is a nontrivial task: on the one hand, letting $\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t, \Psi, \alpha) \neq \bot$ for more values ϕ, t, Ψ, α makes the procedure more likely to succeed; on the other hand, it is computationally more demanding. We explain in Section 6.1 the choice of Ctx made in our implementation.

We write $\Longrightarrow^*$ for the transitive and reflexive closure of $\Longrightarrow$. The definitions of Ctx and of the transformation rules ensure that whenever $S \Longrightarrow^* S'$ and S is a state, then S' is also a state, with the same parameters unless $S' = \bot$.

Example 5. Consider the frame φ_1 previously described in Example 3. We can apply rule **A.1** as follows. Consider the rewrite rule $\mathsf{dec}(\mathsf{enc}(x, y), y) \to x$, the decomposition D_2 given in Example 4 and $t_1 = \mathsf{enc}(\mathsf{c}_0, \mathsf{k})$. We have $\mathrm{Init}(\varphi_1) = (\varphi_1, \emptyset) \Longrightarrow (\varphi_1, \{\mathsf{dec}(\mathsf{w}_1, \mathsf{w}_2) \bowtie \mathsf{c}_0\})$. In other words, since we know the key k through w_2, we can check that the decryption of w_1 by w_2 leads to the public constant c_0. Next we apply rule **B.1** as follows: $(\varphi_1, \{\mathsf{dec}(\mathsf{w}_1, \mathsf{w}_2) \bowtie \mathsf{c}_0\}) \Longrightarrow$

$(\varphi_1, \{\mathsf{dec}(\mathsf{w}_1, \mathsf{w}_2) \bowtie \mathsf{c}_0, \mathsf{enc}(\mathsf{c}_0, \mathsf{w}_2) \bowtie \mathsf{w}_1\})$. No more rules can then modify the state.

Main theorem. We now state the soundness and the completeness of the transformation rules provided that a *saturated state* is reached, that is, a state $S \neq \bot$ such that $S \Longrightarrow S'$ implies $S' = S$. The technical lemmas involved in the proof are detailed in Section 5.

Theorem 1 (soundness and completeness). *Let* E *be an equational theory generated by a convergent rewrite system* $\mathcal{R}$*. Let* φ *be an initial frame and* (Φ, Ψ) *be a saturated state such that* $\mathrm{Init}(\varphi) \Longrightarrow^* (\Phi, \Psi)$*.*

1. *For all* $M \in \mathcal{F}_{\mathsf{pub}}[\mathrm{par}(\varphi)]$ *and* $t \in \mathcal{F}[\emptyset]$*, we have*
$$M\varphi =_{\mathsf{E}} t \quad \Leftrightarrow \quad \exists N,\ \Psi \models M \bowtie N \text{ and } N \rhd_{\Phi} t{\downarrow_{\mathcal{R}}}$$
2. *For all* $M, N \in \mathcal{F}_{\mathsf{pub}}[\mathrm{par}(\varphi) \cup \mathcal{X}]$*, we have that* $M\varphi =_{\mathsf{E}} N\varphi \Leftrightarrow \Psi \models M \bowtie N$*.*

While the saturation procedure is sound and complete, it may not terminate, or *fail* if rule **A.3** becomes the only applicable rule. In Section 6, we explore several sufficient conditions to prevent failure and ensure termination.

4.3 Application to deduction and static equivalence

Decision procedures for deduction and static equivalence follow from Theorem 1.

Algorithm for deduction. Let φ be an initial frame and t be a ground term. The procedure for checking $\varphi \vdash_{\mathsf{E}} t$ runs as follows:

1. Apply the transformation rules to obtain (if any) a saturated state (Φ, Ψ) such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\Phi, \Psi)$;
2. Return *yes* if there exists N such that $N \rhd_{\Phi} t{\downarrow_{\mathcal{R}}}$ (that is, the $\mathcal{R}$-reduced form of t is syntactically deducible from Φ); otherwise return *no.*

Algorithm for static equivalence. Let φ_1 and φ_2 be two initial frames. The procedure for checking $\varphi_1 \approx_{\mathsf{E}} \varphi_2$ runs as follows:

1. Apply the transformation rules to obtain (if possible) two saturated states (Φ_1, Ψ_1) and (Φ_2, Ψ_2) such that $\mathrm{Init}(\varphi_i) \Longrightarrow^* (\Phi_i, \Psi_i)$, $i = 1, 2$;
2. For $\{i, j\} = \{1, 2\}$, for every equation $(\forall z_1, \ldots, z_\ell . M \bowtie N)$ in Ψ_i, check that $M\varphi_j =_{\mathsf{E}} N\varphi_j$ — that is, in other words, $(M\varphi_j){\downarrow_{\mathcal{R}}} = (N\varphi_j){\downarrow_{\mathcal{R}}}$;
3. If so return *yes*; otherwise return *no.*

5 Soundness and completeness of the saturation

The proof of Theorem 1 is based on three main lemmas. First, the transformation rules are sound in the sense that, along the saturation process, we add only deducible terms and valid equations with respect to the initial frame.

Lemma 1 (soundness). *Let φ be an initial frame and (Φ, Ψ) be a state such that* $\mathrm{Init}(\varphi) \Longrightarrow^* (\Phi, \Psi)$*. Then, we have that*

1. $M \rhd_\Phi t \;\Rightarrow\; M\varphi =_{\mathsf{E}} t$ *for all* $M \in \mathcal{F}_{\mathsf{pub}}[\mathrm{dom}(\varphi)]$ *and* $t \in \mathcal{F}[\emptyset]$*;*
2. $\Psi \models M \bowtie N \;\Rightarrow\; M\varphi =_{\mathsf{E}} N\varphi$ *for all* $M, N \in \mathcal{F}_{\mathsf{pub}}[\mathrm{dom}(\varphi) \cup \mathcal{X}]$.

The next two lemmas are dedicated to the completeness of **B** and **A** rules, respectively. Lemma 2 ensures that saturated states account for all the syntactic equations possibly visible. Lemma 3 deals with the reduction of a deducible term along the rewrite system $\mathcal{R}$. Using that $\mathcal{R}$ is convergent, this allows us to prove that every deducible term from a saturated frame is syntactically deducible.

Lemma 2 (completeness, syntactic equations). *Let (Φ, Ψ) be a state, and M, N be two terms such that $M \rhd_\Phi t$ and $N \rhd_\Phi t$ for some term t. Then there exists (Φ', Ψ') such that $(\Phi, \Psi) \Longrightarrow^* (\Phi', \Psi')$ using **B** rules and $\Psi' \models M \bowtie N$.*

Lemma 3 (completeness, context reduction). *Let (Φ, Ψ) be a state and M, t, t' be three terms such that $M \rhd_\Phi t$ and $t \to_{\mathcal{R}} t'$. Then, either $(\Phi, \Psi) \Longrightarrow^* \bot$ or there exist (Φ', Ψ'), M' and t'' such that $(\Phi, \Psi) \Longrightarrow^* (\Phi', \Psi')$, $M' \rhd_{\Phi'} t''$ with $t' \to^*_{\mathcal{R}} t''$, and $\Psi' \models M \bowtie M'$.*

*Besides, in both cases, the corresponding derivation from (Φ, Ψ) can be chosen to consist of a number of **B** rules, possibly followed by one instance of **A** rule involving the same rewrite rule $l \to r$ as the rewrite step $t \to_{\mathcal{R}} t'$.*

6 Termination and non-failure

In the previous section, we proved that saturated frames yield sound and complete characterizations of deducible terms and visible equations of their initial frames. Yet, the saturation procedure may still not terminate, or fail due to rule **A.3**. In this section, we study different conditions on the rewrite system $\mathcal{R}$ so that failure never happens and/or termination is ensured.

6.1 A syntactic criterion to prevent failure

Our first criterion is syntactic and ensures that the algorithm never fails. It is enjoyed by a large class of equational theories, called *layered convergent*.

Definition 4 (layered rewrite system). *A rewrite system $\mathcal{R}$, and by extension its equational theory E, are* layered *if there exists an ascending chain of subsets $\emptyset = \mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \ldots \subseteq \mathcal{R}_{N+1} = \mathcal{R}$ $(N \geq 0)$, such that for every $0 \leq i \leq N$, for every rule $l \to r$ in $\mathcal{R}_{i+1} - \mathcal{R}_i$, for every (n, p, q)-decomposition $l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$, one of the following two conditions holds:*

(i) $\mathrm{var}(r) \subseteq \mathrm{var}(l_1, \ldots, l_n)$*;*

(ii) there exist $C_0, C_1, \ldots, C_k$ and $s_1, \ldots, s_k$ such that

- $r = C_0[s_1, \ldots, s_k]$*;*

- *for each $1 \leq i \leq k$, $C_i[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$ rewrites to s_i in zero or one step of rewrite rule in head position along $\mathcal{R}_i$.*

In the latter case, we say that the context $C = C_0[C_1, \ldots, C_k]$ is associated *to the decomposition D of $l \to r$. Note that $C[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q] \to^*_{\mathcal{R}_i} r$.*

Proposition 1. *Assume that the function* Ctx *in use is* maximal*: for every ϕ and t, if there exists s such that $\phi \vdash s$ and $t \to^*_{\mathcal{R}} s$, then for any Ψ, α,* $\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t, \Psi, \alpha) \neq \bot$*. Then, provided that $\mathcal{R}$ is layered convergent, there exists no state (Φ, Ψ) from which $(\Phi, \Psi) \Longrightarrow \bot$ is the only applicable derivation.*

Practical considerations. Unfortunately, such a maximal Ctx is too inefficient in practice as one has to consider the syntactic deducibility problem $\phi \vdash^? s$ for every $t \to^*_{\mathcal{R}} s$. This is why we rather use the following lighter implementation:

- for every index $0 \leq i \leq N$, and every rule $l \to r$ in $\mathcal{R}_{i+1} - \mathcal{R}_i$, if $l = D[l_1, \ldots, l_n, y_1, \ldots, y_{p+q}]$ is a (n, p, q)-decomposition satisfying condition (ii) above for some (arbitrarily chosen) associated context C, then, for every ϕ and σ such that $\phi \vdash l\sigma$, we let

$$\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} r\sigma,\, \Psi,\, (l, r, D, \sigma)) \;=\; C[M_1, \ldots, M_{n+p+q}]$$

  where the M_k are fixed recipes such that $(M_i \rhd l_i\sigma) \in \phi$ for $1 \leq i \leq n$ and $(M_{n+j} \rhd y_j\sigma) \in \phi$ for $1 \leq j \leq p+q$;
- otherwise, if $\phi \vdash t{\downarrow_{\mathcal{R}}}$, we let $\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t, \Psi, \alpha)$ be some fixed M such that $M \rhd_\phi t{\downarrow_{\mathcal{R}}}$;
- in any other case, we let $\mathrm{Ctx}(\phi \vdash^?_{\mathcal{R}} t, \Psi, \alpha) \;=\; \bot$.

Using similar ideas as for the proof of Proposition 1, we can show that, for any convergent rewrite system $\mathcal{R}$, this choice of Ctx is compatible with property (b) of Subsection 4.2, and more generally with completeness, as long as, during saturation, the transformation rules **A** involve the rewrite rules of $\mathcal{R}_i$ with greater priority than those of $\mathcal{R}_j$, $i < j$. Moreover, when $\mathcal{R}$ is additionally layered, this definition ensures that the procedure never fails. Indeed, using the notations of Figure 1, $\mathrm{Ctx}(\Phi \cup \{z_1 \rhd z_1, \ldots, z_q \rhd z_q\} \vdash^?_{\mathcal{R}} r\sigma,\, \Psi,\, (l, r, D, \sigma)) \;=\; \bot$ implies that (ii) is false on D, thus (i) $\mathrm{var}(r) \subseteq \mathrm{var}(l_1, \ldots, l_n)$ holds and $(r\sigma){\downarrow_{\mathcal{R}}}$ is ground.

Example 6. Any convergent subterm rewrite system $\mathcal{R}$ is layered convergent. Indeed, let $N = 0$ and $\mathcal{R}_1 = \mathcal{R}$. For any $l \to r$ in $\mathcal{R}$ and for every decomposition $l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$, the term r is a subterm of l, thus either $r = C[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$ for some context C, or r is a subterm of some l_i thus $\mathrm{var}(r) \subseteq \mathrm{var}(l_1, \ldots, l_n)$.

Example 7. Other examples are provided by the theory of homomorphism $\mathsf{E}_{\mathsf{hom}}$ defined in Section 2.3 as well as the convergent theories of blind signatures $\mathsf{E}_{\mathsf{blind}}$ and prefix encryption $\mathsf{E}_{\mathsf{pref}}$ defined by the following sets of equations.

$$\mathcal{E}_{\mathsf{blind}} = \mathcal{E}_{\mathsf{enc}} \cup \left\{ \begin{array}{c} \mathsf{unblind}(\mathsf{blind}(x, y), y) = x \\ \mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x, y), z), y) = \mathsf{sign}(x, z) \end{array} \right\}$$

$$\mathcal{E}_{\mathsf{pref}} = \mathcal{E}_{\mathsf{enc}} \cup \left\{ \mathsf{pref}(\mathsf{enc}(\langle x, y\rangle, z)) = \mathsf{enc}(x, z) \right\}$$

The theory $\mathsf{E}_{\mathsf{blind}}$ models primitives used in e-voting protocols [17]. The prefix theory represents the property of many chained modes of encryption (e.g. CBC) where an attacker can retrieve any encrypted prefix out of a ciphertext.

Let us check for instance that the prefix theory $\mathsf{E}_{\mathsf{pref}}$ is layered. Let $N = 1$, $\mathcal{R}_1$ be the rewrite system obtained from $\mathcal{E}_{\mathsf{enc}}$ by orienting the equations from left to right, and $\mathcal{R}_2 = \mathcal{R}_1 \cup \{\mathsf{pref}(\mathsf{enc}(\langle x, y\rangle, z)) \to \mathsf{enc}(x, z)\}$. The rewrite rules of $\mathcal{R}_1$ satisfy the assumptions since $\mathcal{R}_1$ forms a convergent subterm rewrite system. The additional rule $\mathsf{pref}(\mathsf{enc}(\langle x, y\rangle, z)) \to \mathsf{enc}(x, z)$ admits three decompositions up to permutation of parameters:

- $l = \mathsf{pref}(l_1)$, in which case $\mathrm{var}(r) \subseteq \mathrm{var}(l_1)$;
- $l = \mathsf{pref}(\mathsf{enc}(l_1, z))$, in which case $\mathsf{enc}(\pi_1(l_1), z) \to_{\mathcal{R}_1} r$;
- $l = \mathsf{pref}(\mathsf{enc}(\langle x, y\rangle, z))$, in which case $r = \mathsf{enc}(x, z)$.

Verifying that the convergent theories $\mathsf{E}_{\mathsf{hom}}$ and $\mathsf{E}_{\mathsf{blind}}$ are layered is similar.

6.2 Termination

In the previous subsection, we described a sufficient criterion for non-failure. To obtain decidability for a given layered convergent theory, there remains only to provide a termination argument. Such an argument is generally easy to develop by hand as we illustrate on the example of the prefix theory. For the case of existing decidability results from [2], such as the theories of blind signature and homomorphic encryption, we also provide a semantic criterion that allows us to directly conclude termination of the procedure.

Proving termination by hand. To begin with, we note that **B** rules always terminate after a polynomial number of steps. Let us write $\overset{\centerdot}{\Longrightarrow}^n$ for the relation made of exactly n *strict applications* of rules ($S \overset{\centerdot}{\Longrightarrow} S'$ iff $S \Longrightarrow S'$ and $S \neq S'$).

Proposition 2. *For every states $S = (\Phi, \Psi)$ and S' such that $S \overset{\centerdot}{\Longrightarrow}^n S'$ using only **B** rules, n is polynomially bounded in the size of* $\mathrm{im}(\Phi)$.

This is due to the fact that frames are one-to-one and that the rule **B.2** only adds deduction facts $M \rhd t$ such that t is a subterm of an existing term in Φ. Hence, for proving termination, we observe that it is sufficient to provide a function s mapping each frame Φ to a finite set of terms $s(\Phi)$ including the subterms of $\mathrm{im}(\Phi)$ and such that rule **A.2** only adds deduction facts $M \rhd t$ satisfying $t \in s(\Phi)$.

For subterm theories, we obtain polynomial termination by choosing $s(\Phi)$ to be the subterms of $\mathrm{im}(\Phi)$ together with the ground right-hand sides of $\mathcal{R}$.

Proposition 3. *Let E be a convergent subterm theory. For every $S = (\Phi, \Psi)$ and S' such that $S \overset{\centerdot}{\Longrightarrow}^n S'$, n is polynomially bounded in the size of* $\mathrm{im}(\Phi)$.

To conclude that deduction and static equivalence are decidable in polynomial time [2], we need to show that the deduction facts and the equations are of polynomial size. This requires a DAG representation for terms and visible equations. For our implementation, we have chosen not to use DAGs for the sake of simplicity (and perhaps efficiency) since DAGs require much heavier data structures. However, similar techniques as those described in [2] would apply to implement our procedure using DAGs.

For proving termination of the prefix theory, we let $s(\Phi)$ be the minimal set containing Φ, closed by subterm and such that $\mathsf{enc}(t_1, k) \in s(\Phi)$ whenever $\mathsf{enc}(\langle t_1, t_2\rangle, k) \in s(\Phi)$. We then deduce that deduction and static equivalence are decidable for the equational theory $\mathsf{E}_{\mathsf{pref}}$, which is a new decidability result.

A criterion to ensure termination. We now provide a semantic criterion that more generally explains why our procedure succeeds on theories previously known to be decidable [2]. This criterion intuitively states that the set of deducible terms from any initial frame φ should be equivalent to a set of *syntactically* deducible terms. Provided that failures are prevented and assuming a *fair* strategy for rule application, we prove that this criterion is a necessary and sufficient condition for our procedure to terminate.

Definition 5 (fair derivation). *An infinite derivation* $(\Phi_0, \Psi_0) \Longrightarrow \ldots \Longrightarrow (\Phi_n, \Psi_n) \Longrightarrow \ldots$ *is* fair *iff along this derivation,*

(a) ***B*** *rules are applied with greatest priority, and*
(b) whenever a ***A*** *rule is applicable for some instance* $(l \to r, D, t_1, \ldots, t_n, \ldots)$*, eventually the same instance of rule is applied during the derivation.*

Fairness implies that any deducible term is eventually syntactically deducible.

Lemma 4. *Let* $S_0 = (\Phi_0, \Psi_0) \Longrightarrow \ldots \Longrightarrow (\Phi_n, \Psi_n) \Longrightarrow \ldots$ *be an infinite fair derivation from a state* S_0*. For every ground term* t *such that* $\Phi_0 \vdash_{\mathsf{E}} t$*, either* $(\Phi_0, \Psi_0) \Longrightarrow^* \bot$ *or there exists* i *such that* $\Phi_i \vdash t{\downarrow_{\mathcal{R}}}$*.*

Proposition 4 (criterion for saturation). *Let* φ *be an initial frame such that* $\mathrm{Init}(\varphi) \not\Longrightarrow^* \bot$*. The following conditions are equivalent:*

(i) There exists a saturated couple (Φ, Ψ) *such that* $\mathrm{Init}(\varphi) \Longrightarrow^* (\Phi, \Psi)$*.*
(ii) There exists a (finite) initial frame φ_s *such that for every term* t*,* t *is deducible from* φ *modulo* E *iff* $t{\downarrow_{\mathcal{R}}}$ *is syntactically deducible from* φ_s*.*
(iii) There exists no fair infinite derivation starting from $\mathrm{Init}(\varphi)$*.*

Together with the syntactic criterion described in Section 6.1, this criterion (Property *(ii)*) allows us to prove decidability of deduction and static equivalence for layered convergent theories that belong to the class of *locally stable* theories defined in [2]. As a consequence, our procedure always saturates for the theories of blind signatures and homomorphic encryption since those theories are layered and have been proved locally stable [2]. Other examples of layered convergent theories enjoying this criterion can be found in [2] (e.g. a theory of addition).

7 Implementation: the YAPA tool

YAPA is an Ocaml implementation[4] of the saturation procedure presented in Section 4, using by default the optimized function Ctx defined in Section 6, and a fair strategy of rule application (see Definition 5).

The tool takes as input an equational theory described by a finite convergent rewrite system, as well as frame definitions and queries. A few optimizations may be activated for subterm theories, e.g. to accelerate normalization. The procedure starts by computing the decompositions of the rewrite system. Provided that the rewrite rules are given in an order compatible with the sets $\mathcal{R}_0 \subseteq \ldots \subseteq \mathcal{R}_{N+1}$ of Definition 4, it is able to recognize (fully or partially) layered theories and to pre-compute the associated contexts C related to condition (ii) of this definition, and exploited by the function Ctx in use for eliminating failure cases.

We have conducted several experiments on a PC Intel Core 2 Duo at 2.4 GHz with 2 Go RAM for various equational theories (see below) and found that YAPA provides an efficient way to check static equivalence and deducibility.

Equational theory	$\mathsf{E_{enc}}$ $n = 10$	$\mathsf{E_{enc}}$ $n = 14$	$\mathsf{E_{enc}}$ $n = 16$	$\mathsf{E_{enc}}$ $n = 18$	$\mathsf{E_{enc}}$ $n = 20$	$\mathsf{E_{blind}}$	$\mathsf{E_{pref}}$	$\mathsf{E_{hom}}$	$\mathsf{E_{add}}$
Execution time	< 1s	1,7s	8s	30s	< 3min	< 1s	< 1s	< 1s	< 1s

For the case of $\mathsf{E_{enc}}$, we have run YAPA on the frames $\varphi_n = \{\mathsf{w}_1 \rhd t_n^0, \mathsf{w}_2 \rhd \mathsf{c}_0, \mathsf{w}_3 \rhd \mathsf{c}_1\}$ and $\varphi'_n = \{\mathsf{w}_1 \rhd t_n^1, \mathsf{w}_2 \rhd \mathsf{c}_0, \mathsf{w}_3 \rhd \mathsf{c}_1\}$, where $t_0^i = \mathsf{c}_i$ and $t_{n+1}^i = \langle \mathsf{enc}(t_n^i, \mathsf{k}_n^i), \mathsf{k}_n^i \rangle$, $i \in \{0, 1\}$. These examples allow us to increase the (tree, non-DAG) size of the distinguishing tests exponentially, while the sizes of the frames grow linearly. Despite the size of the output, we have observed satisfactory performances for the tool. We have also experimented YAPA on several convergent theories, e.g. $\mathsf{E_{blind}}$, $\mathsf{E_{hom}}$, $\mathsf{E_{pref}}$ and the theory of addition $\mathsf{E_{add}}$ defined in [2].

In comparison with the tool ProVerif [9, 10], here instrumented to check static equivalences, our test samples suggest a running time between one and two orders of magnitude faster for YAPA. Also we did not succeed in making ProVerif terminate on the two theories $\mathsf{E_{hom}}$ and $\mathsf{E_{add}}$. Of course, these results are not entirely surprising given that ProVerif is tailored for the more general (and difficult) problem of protocol (in)security under active adversaries. In particular ProVerif's initial preprocessing of the rewrite system appears more substantial than ours and does not terminate on the theories $\mathsf{E_{hom}}$ and $\mathsf{E_{add}}$ (although termination is guaranteed for linear or subterm-convergent theories [10]).

Altogether, these results suggest that YAPA significantly improves the state of the art for checking deducibility and static equivalence under convergent theories, both from practical and theoretical perspectives.

References

1. M. Abadi, M. Baudet, and B. Warinschi. Guessing attacks and the computational soundness of static equivalence. In *Foundations of Software Science and Computation Structures (FOSSACS'06)*, pages 398–412, 2006.

[4] Freely available at `http://www.lsv.ens-cachan.fr/~baudet/yapa/`

2. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, 2006.
3. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM, 2001.
4. S. Anantharaman, P. Narendran, and M. Rusinowitch. Intruders with caps. In *18th International Conference on Term Rewriting and Applications (RTA'07)*, volume 4533 of *LNCS*. Springer, 2007.
5. M. Baudet. Deciding security of protocols against off-line guessing attacks. In *12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM Press, 2005.
6. M. Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. Thèse de doctorat, LSV, ENS Cachan, France, 2007.
7. M. Baudet, V. Cortier, and S. Delaune. YAPA: A generic tool for computing intruder knowledge. Research Report LSV-09-03, Laboratoire Spécification et Vérification, ENS Cachan, France, Feb. 2009. 28 pages.
8. M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In *32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *LNCS*, pages 652–663. Springer, 2005.
9. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Comp. Soc. Press, 2001.
10. 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.
11. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *18th IEEE Symposium on Logic in Computer Science (LICS'03)*. IEEE Comp. Soc. Press, 2003.
12. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *18th IEEE Symposium on Logic in Computer Science (LICS'03)*. IEEE Comp. Soc. Press, 2003.
13. R. Corin, J. Doumen, and S. Etalle. Analysing password protocol security against off-line dictionary attacks. In *2nd International Workshop on Security Issues with Petri Nets and other Computational Models (WISP'04)*, ENTCS, 2004.
14. V. Cortier and S. Delaune. Deciding knowledge in security protocols for monoidal equational theories. In *14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, LNAI. Springer, 2007.
15. V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.
16. S. Delaune and F. Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In *11th ACM Conference on Computer and Communications Security (CCS'04)*, pages 278–287, 2004.
17. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 2008. To appear.
18. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, 1996.
19. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *8th ACM Conference on Computer and Communications Security (CCS'01)*, 2001.

Journal of Automated Reasoning manuscript No.
(will be inserted by the editor)

Computing knowledge in security protocols under convergent equational theories

Ştefan Ciobâcă · Stéphanie Delaune · Steve Kremer

the date of receipt and acceptance should be inserted later

Abstract The analysis of security protocols requires reasoning about the knowledge an attacker acquires by eavesdropping on network traffic. In formal approaches, the messages exchanged over the network are modeled by a term algebra equipped with an equational theory axiomatizing the properties of the cryptographic primitives (e.g. encryption, signature). In this context, two classical notions of knowledge, deducibility and indistinguishability, yield corresponding decision problems.

We propose a procedure for both problems under arbitrary convergent equational theories. Since the underlying problems are undecidable we cannot guarantee termination. Nevertheless, our procedure terminates on a wide range of equational theories. In particular, we obtain a new decidability result for a theory we encountered when studying electronic voting protocols. We also provide a prototype implementation.

Keywords Formal methods, security protocols, equational theories, static equivalence.

1 Introduction

Cryptographic protocols are small distributed programs that use cryptographic primitives such as encryption and digital signatures to communicate securely over a network. It is essential to gain as much confidence as possible in their correctness. Therefore, symbolic methods have been developed to analyse such protocols [4, 24, 26]. In these approaches, one of the most important aspects is to be able to reason about the *knowledge* of the attacker.

Traditionally, the knowledge of the attacker is expressed in terms of *deducibility* (e.g. [26, 14]). A message s (intuitively the secret) is said to be deducible from a set of messages φ, if an attacker is able to compute s from φ. To perform this computation, the attacker is allowed, for example, to decrypt deducible messages by deducible keys.

This work has been partly supported by the ANR SeSur project AVOTÉ. A preliminary version of this work was presented in [17].

LSV, ENS Cachan & CNRS & INRIA, France E-mail: { ciobaca | delaune | kremer } @ lsv.ens-cachan.fr

However, deducibility is not always sufficient. Consider for example the case where a protocol participant sends over the network the encryption of one of the constants "yes" or "no" (e.g. the value of a vote). Deducibility is not the right notion of knowledge in this case, since both possible values ("yes" and "no") are indeed "known" to the attacker. In this case, a more adequate form of knowledge is *indistinguishability* (e.g. [1]): is the attacker able to distinguish between two transcripts of the protocol, one running with the value "yes" and the other one running with the value "no"?

In symbolic approaches to cryptographic protocol analysis, the protocol messages and cryptographic primitives (e.g. encryption) are generally modeled using a term algebra. This term algebra is interpreted modulo an equational theory. Using equational theories provides a convenient and flexible framework for modeling cryptographic primitives [20]. For instance, a simple equational theory for symmetric encryption can be specified by the equation $dec(enc(x, y), y) = x$. This equation models the fact that decryption cancels out encryption when the same key is used. Different equational theories can also be used to model randomized encryption or even more complex primitives arising when studying electronic voting protocols [21,6] or direct anonymous attestation [7]: blind signatures, trapdoor commitments, zero-knowledge proofs, ...

The two notions of knowledge that we consider do not take into account the dynamic behaviour of the protocol. Nevertheless, in order to establish that two dynamic behaviors of a protocol are indistinguishable, an important subproblem is to establish indistinguishability between the sequences of messages generated by the protocol [26, 2]. Indistinguishability, also called static equivalence in the applied-pi calculus framework [2], plays an important role in the study of guessing attacks (e.g. [18,8]), as well as for anonymity properties in e-voting protocols (e.g. [21,6]). This was actually the starting point of this work. During the study of e-voting protocols, we came across several equational theories for which we needed to show static equivalence while no decision procedure for deduction or static equivalence existed.

*Our contributions.*We provide a procedure which is correct, in the sense that if it terminates it gives the right answer, for any convergent equational theory. As deduction and static equivalence are undecidable for this class of equational theories [1], the procedure does not always terminate. However, we show that it does terminate for the class of *subterm convergent* equational theories (already shown decidable in [1]) and several other theories among which the theory of *trapdoor commitment* encountered in our electronic voting case studies [21].

Our second contribution is an efficient prototype implementation of this generic procedure. Our procedure relies on a simple fixed point computation based on a few saturation rules, making it convenient to implement.

*Related work.*Many decision procedures have been proposed for deducibility (e.g. [14,3, 23,15]) under a variety of equational theories modeling encryption, digital signatures, exclusive OR, and homomorphic operators. Several papers are also devoted to the study of static equivalence. Most of these results introduce a new procedure for each particular theory and even in the case of the general decidability criterion given in [1, 19], the algorithm underlying the proof has to be adapted for each particular theory, depending on how the criterion is fulfilled. A combination result was obtained in [5]: if deduction (and resp. static equivalence) is decidable for two disjoint equational theories, then deduction (and resp. static equivalence) is decidable for the union of the two theories.

The first generic algorithm that has been proposed handles subterm convergent equational theories [1] and covers the classical theories for encryption and signatures. This result is encompassed by the recent work of Baudet *et al.* [10] in which the authors propose a generic procedure that works for any convergent equational theory, but which may fail or not terminate. This procedure has been implemented in the YAPA tool [9] and has been shown to terminate without failure in several cases (e.g. subterm convergent theories and blind signatures). However, due to its simple representation of deducible terms (represented by a finite set of *ground* terms), the procedure fails on several interesting equational theories like the theory of trapdoor commitments. Our representation of deducible terms overcomes this limitation by including terms with variables which can be substituted by any deducible terms. Independently of our work, specific decision procedures for the theory of trapdoor commitment and that of reencryption have been presented in [11].

Another tool that can be used to check static equivalence is ProVerif [12,13]. This tool can handle various equational theories and analyse security protocols under active adversaries. However, termination is not guaranteed in general and the tool perform some safe approximations.

2 Formal model

2.1 Term algebras

As usual, messages will be modeled using a term algebra. Let $\mathcal{F}$ be a finite set of *function symbols* coming with an arity function $\text{ar} : \mathcal{F} \to \mathbb{N}$. Function symbols of arity 0 are called *constants*. We consider several kind of *atoms* among which an infinite set of *names* $\mathcal{N}$, an infinite set of *variables* $\mathcal{X}$ and a set of parameters $\mathcal{P}$. The set of terms $\mathcal{T}(\mathcal{F}, \mathcal{A})$ built over $\mathcal{F}$ and the atoms in $\mathcal{A}$ is defined as

$$\begin{array}{rlll} t, t_1, \ldots ::= & & \text{term} & \\ & \mid \; a & & \text{atom } a \in \mathcal{A} \\ & \mid \; f(t_1, \ldots, t_k) & & \text{application of symbol } f \in \mathcal{F}, \text{ar}(f) = k \end{array}$$

A term t is said to be *ground* when $t \in \mathcal{T}(\mathcal{F}, \mathcal{N})$. We assume the usual definitions to manipulate terms. We write $\text{fn}(t)$ (resp. $\text{var}(t)$) the set of (free) names (resp. variables) that occur in a term t and $\text{st}(t)$ the set of its (syntactic) subterms. These notations are extended to tuples and sets of terms in the usual way. We denote by $|t|$ the *size* of t defined as the number of symbols that occur in t (variables do not count), and $\#T$ denotes the *cardinality* of the set T.

The set of positions of a term t is written $\text{pos}(t) \subseteq \mathbb{N}^*$. If p is a position of t then $t|_p$ denotes the subterm of t at the position p. The term $t[u]_p$ is obtained from t by replacing the occurrence of $t|_p$ at position p with u. A *context* C is a term with (1 or more) holes and we write $C[t_1, \ldots t_n]$ for the term obtained by replacing these holes with the terms $t_1, \ldots, t_n$. A context is *public* if it only consists of function symbols and holes.

Substitutions are written $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ with $\text{dom}(\sigma) = \{x_1, \ldots, x_n\}$. The application of a substitution σ to a term t is written $t\sigma$. The substitution σ is *grounding for* $t_1, \ldots, t_k$ if the resulting terms $t_1\sigma, \ldots, t_k\sigma$ are ground. We use the same notations for *replacements* of names and parameters by terms.

2.2 Equational theories and rewriting systems

Equality between terms will generally be interpreted modulo an *equational theory*. An equational theory $\mathcal{E}$ is defined by a set of equations $M \sim N$ with $M, N \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Equality modulo $\mathcal{E}$, written $=_{\mathcal{E}}$, is defined to be the smallest equivalence relation on terms such that $M =_{\mathcal{E}} N$ for all $M \sim N \in \mathcal{E}$ and which is closed under substitution of terms for variables and application of contexts.

It is often more convenient to manipulate rewriting systems than equational theories. A *rewriting system* $\mathcal{R}$ is a set of rewriting rules $l \to r$ where $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $\mathrm{var}(r) \subseteq \mathrm{var}(l)$. A term t rewrites to t' by $\mathcal{R}$, denoted by $t \to_{\mathcal{R}} t'$, if there exist $l \to r \in \mathcal{R}$, a position $p \in \mathrm{pos}(t)$ and a substitution σ such that $t|_p = l\sigma$ and $t' = t[r\sigma]_p$. We denote by $\to_{\mathcal{R}}^{+}$ the transitive closure of $\to_{\mathcal{R}}$, $\to_{\mathcal{R}}^{*}$ its reflexive and transitive closure, and $=_{\mathcal{R}}$ its reflexive, symmetric and transitive closure.

A rewrite system $\mathcal{R}$ is *convergent* if is *terminating*, i.e. there is no infinite chain $u_1 \to_{\mathcal{R}} u_2 \to_{\mathcal{R}} \ldots$, and *confluent*, i.e. for every terms u_1, u_2 such that $u_1 =_{\mathcal{R}} u_2$, there exists u such that $u_1 \to_{\mathcal{R}}^{*} u$ and $u_2 \to_{\mathcal{R}}^{*} u$. A term u is in $\mathcal{R}$*-normal form* if there is no term u' such that $u \to_{\mathcal{R}} u'$. If $u \to_{\mathcal{R}}^{*} u'$ and u' is in $\mathcal{R}$-normal form then u' is *an* $\mathcal{R}$*-normal form of* u. When this reduced form is unique (in particular if $\mathcal{R}$ is convergent), we write $u' = u{\downarrow}_{\mathcal{R}_{\mathcal{E}}}$.

We are particularly interested in theories $\mathcal{E}$ that can be represented by a convergent rewrite system $\mathcal{R}$, i.e. theories for which there exists a convergent rewrite system $\mathcal{R}$ such that the two relations $=_{\mathcal{R}}$ and $=_{\mathcal{E}}$ coincide. Given an equational theory $\mathcal{E}$ we define the corresponding rewriting system $\mathcal{R}_{\mathcal{E}}$ by orienting all equations in $\mathcal{E}$ from left to right, i.e., $\mathcal{R}_{\mathcal{E}} = \{l \to r \mid l \sim r \in \mathcal{E}\}$. We say that $\mathcal{E}$ is *convergent* if $\mathcal{R}_{\mathcal{E}}$ is convergent.

Example 1 A classical equational theory modelling symmetric encryption is $\mathcal{E}_{enc} = \{dec(enc(x, y), y) \sim x\}$. As a running example we consider a slight extension of this theory modelling *malleable* encryption

$$\mathcal{E}_{mal} = \mathcal{E}_{enc} \cup \{mal(enc(x, y), z) \sim enc(z, y)\}.$$

This malleable encryption scheme allows one to arbitrarily change the plaintext of an encryption. This theory certainly does not model a realistic encryption scheme but it yields a simple example of a theory which illustrates well our procedures. In particular all existing decision procedure we are aware of fail on this example. The rewriting system $\mathcal{R}_{\mathcal{E}_{mal}}$ is convergent.

From now on, assume we are given a convergent equational theory $\mathcal{E}$ built over a signature $\mathcal{F}$ and represented by the convergent rewriting system $\mathcal{R}_{\mathcal{E}}$.

2.3 Deducibility and static equivalence

In order to describe the messages observed by an attacker, we consider the following notion of *frame* that comes from the applied-pi calculus [2].

A frame φ is a sequence of messages $u_1, \ldots, u_n$ meaning that the attacker observed each of these messages in the given order. Furthermore, we distinguish the names that the attacker knows from those that were freshly generated by others and that are *a priori* unknown by the attacker. Formally, a frame φ is defined as $\nu\tilde{n}.\sigma$ where $\tilde{n}$ is its set of bound names, denoted by $\mathrm{bn}(\varphi)$, and a replacement $\sigma = \{w_1 \mapsto u_1, \ldots, w_n \mapsto u_n\}$.

The parameters $w_1, \ldots, w_n$ enable us to refer to $u_1, \ldots, u_n \in \mathcal{T}(\mathcal{F}, \mathcal{N})$. The *domain* $\text{dom}(\varphi)$ of φ is $\{w_1, \ldots, w_n\}$.

Let $\varphi = \nu\tilde{n}.\sigma$. Given terms M and N such that $\text{fn}(M, N) \cap \tilde{n} = \emptyset$, we sometimes write $(M =_\mathcal{E} N)\varphi$ (resp. $M\varphi$) instead of $M\sigma =_\mathcal{E} N\sigma$ (resp. $M\sigma$).

Definition 1 (deducibility) Let φ be a frame. A ground term t is *deducible in $\mathcal{E}$ from φ*, written $\varphi \vdash_\mathcal{E} t$, if there exists $M \in \mathcal{T}(\mathcal{F}, \mathcal{N} \cup \text{dom}(\varphi))$, called the *recipe*, such that $\text{fn}(M) \cap \text{bn}(\varphi) = \emptyset$ and $M\varphi =_\mathcal{E} t$.

Deducibility does not always suffice for expressing the knowledge of an attacker. This notion does not allow one to express indistinguishability between two sequences of messages. Sometimes, the attacker can deduce the same set of terms from two different frames but he could still be able to distinguish these two frames. This motivates the following notion of static equivalence introduced in [2].

Definition 2 (static equivalence) Let φ_1 and φ_2 be two frames such that $\text{bn}(\varphi_1) = \text{bn}(\varphi_2)$. They are *statically equivalent* in $\mathcal{E}$, written $\varphi_1 \approx_\mathcal{E} \varphi_2$, if

- $\text{dom}(\varphi_1) = \text{dom}(\varphi_2)$
- for all terms $M, N \in \mathcal{T}(\mathcal{F}, \mathcal{N} \cup \text{dom}(\varphi_1))$ such that $\text{fn}(M, N) \cap \text{bn}(\varphi_1) = \emptyset$

$$(M =_\mathcal{E} N)\varphi_1 \;\Leftrightarrow\; (M =_\mathcal{E} N)\varphi_2.$$

Example 2 Consider the two frames described below:

$$\varphi_1 = \nu a, k.\{w_1 \mapsto \mathit{enc}(a, k)\} \quad \text{and} \quad \varphi_2 = \nu a, k.\{w_1 \mapsto \mathit{enc}(b, k)\}.$$

We have that b and $\mathit{enc}(c, k)$ are deducible from φ_2 in $\mathcal{E}_{mal}$ with recipes b and $\mathit{mal}(w_1, c)$ respectively. We have that $\varphi_1 \not\approx_{\mathcal{E}_{mal}} \varphi_2$ since $(w_1 \neq_{\mathcal{E}_{mal}} \mathit{mal}(w_1, b))\varphi_1$ while $(w_1 =_{\mathcal{E}_{mal}} \mathit{mal}(w_1, b))\varphi_2$. Note that $\varphi_1 \approx_{\mathcal{E}_{enc}} \varphi_2$ (in the theory $\mathcal{E}_{enc}$).

3 Procedures for deduction and static equivalence

In this section we describe our procedures for checking deducibility and static equivalence on convergent equational theories. After some preliminary definitions, we present the main part of our procedure, i.e. a set of saturation rules used to reach a fixed point. Then, we show how to use this saturation procedure to decide deducibility and static equivalence. Soundness and completeness of the saturation procedure are stated in Theorem 1 and detailed in Section 4.

Since both problems are undecidable for arbitrary convergent equational theories [1], our saturation procedure does not always terminate. In Section 5, we exhibit (classes of) equational theories for which the saturation terminates.

3.1 Preliminary definitions

We consider two binary predicates $\rhd$ and $\sim$ on terms, which we write using infix notation. These predicates are interpreted over frames φ as follows:

1. $R \rhd t$ is true whenever R is a recipe for t in φ
2. $U \sim V$ whenever $(U =_\mathcal{E} V)\varphi$

The main data structures of our algorithm are two types of Horn clauses, written in this paper as $[H \mid \{L_1, \ldots, L_n\}]$ (read as $L_1 \wedge \ldots \wedge L_n$ implies H), which we call *deduction facts* and respectively *equational facts*.

Definition 3 (facts) A *deduction fact* (resp. an *equational fact*) is an expression denoted $[U \rhd u \mid \Delta]$ (resp. $[U \sim V|\Delta]$) where Δ is a finite set of the form $\{X_1 \rhd t_1, \ldots, X_n \rhd t_n\}$ that contains the *side conditions* of the fact. Moreover, we assume that:

– $u, t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathcal{X})$ with $\mathrm{var}(u) \subseteq \mathrm{var}(t_1, \ldots, t_n)$;
– $U, V \in \mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathcal{X} \cup \mathcal{P})$ and $X_1, \ldots, X_n$ are distinct variables;
– $\mathrm{var}(U, V, X_1, \ldots, X_n) \cap \mathrm{var}(u, t_1, \ldots, t_n) = \emptyset$.

A fact is *solved* if $t_i \in \mathcal{X}$ $(1 \leq i \leq k)$. Otherwise, it is *unsolved.* A deduction fact is *well-formed* if it is unsolved or if $u \notin \mathcal{X}$.

For notational convenience we sometimes omit curly braces for the set of side conditions and write $[U \rhd u \mid X_1 \rhd t_1, \ldots, X_n \rhd t_n]$. When $n = 0$ we simply write $[U \rhd u]$ or $[U \sim V]$.

We say that two facts are equivalent if they are equal up to bijective renaming of variables. In the following we implicitly suppose that all operations are carried out modulo the equivalence classes. In particular set union will not add equivalent facts and inclusion will test for equivalent facts. Also, we allow *on-the-fly* renaming of variables in facts to avoid variable clashes.

We now introduce the notion of generation of a term t from a set of facts F. A term t is generated with recipe R from a set of facts F if $R \rhd t$ is a consequence of the solved facts in F. Formally, we have:

Definition 4 (generation) Let F be a finite set of well-formed deduction facts. A term t is *generated by* F *with recipe* R, written $\mathsf{F} \vdash^R t$, if

1. either $t = x \in \mathcal{X}$ and $R = x$;
2. or there exist a solved fact $[R_0 \rhd t_0 \mid X_1 \rhd x_1, \ldots, X_n \rhd x_n] \in \mathsf{F}$, some terms R_i for $1 \leq i \leq n$ and a substitution σ with $\mathrm{dom}(\sigma) \subseteq \mathrm{var}(t_0)$ such that $t = t_0\sigma$, $R = R_0[X_1 \mapsto R_1, \ldots, X_n \mapsto R_n]$, and $\mathsf{F} \vdash^{R_i} x_i\sigma$ for every $1 \leq i \leq n$.

A term t is *generated by* F, written $\mathsf{F} \vdash t$, if there exists R such that $\mathsf{F} \vdash^R t$.

From this definition follows a simple recursive algorithm for effectively deciding whether $\mathsf{F} \vdash t$, providing also the recipe. Termination is ensured by the fact that $|x_i\sigma| < |t|$ for every $1 \leq i \leq n$. Note that using memoization we can obtain an algorithm in polynomial time.

Example 3 Consider the following set of facts:

$$\begin{array}{lccccl}
[& w_1 & \rhd & enc(b,k) & \mid \emptyset] & (\mathsf{f}_1)\\
[& b & \rhd & b & \mid \emptyset] & (\mathsf{f}_2)\\
[& enc(Y_1,Y_2) & \rhd & enc(y_1,y_2) & \mid Y_1 \rhd y_1, Y_2 \rhd y_2] & (\mathsf{f}_3)
\end{array}$$

where w_1 is a parameter, a, b, k are names, and Y_1, Y_2, y_1, y_2 are variables. We have that $enc(enc(b,k), b)$ is generated with recipe $enc(w_1, b)$. This follows easily by instantiating the two side conditions of f_3 with f_1 and respectively f_2.

Given a finite set of equational facts E and terms M, N, we write $\mathsf{E} \models M \sim N$ if $M \sim N$ is a consequence, in the usual first order theory of equality, of

$$\{U\sigma \sim V\sigma \mid [U \sim V \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k] \in \mathsf{E}\} \text{ where } \sigma = \{X_i \mapsto x_i\}_{1 \leq i \leq k}.$$

Note that it may be the case that $x_i = x_j$ for $i \neq j$ (whereas $X_i \neq X_j$).

3.2 Saturation procedure

We define for each fact f its *canonical form* f' which is obtained by first applying Rule (1) as much as possible and then Rule (2) as much as possible. The idea is to ensure that each variable x_i occurs at most once in the side conditions and to get rid of those variables that do not occur in t. This will be particularly useful to caracterize the form of solved facts when we prove termination in Section 5. Unsolved deduction facts are kept unchanged.

$$(1) \quad \frac{[R \rhd t \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k] \;\; \{i, j\} \subseteq \{1, \ldots, n\} \;\; j \neq i \text{ and } x_j = x_i}{[R\{X_i \mapsto X_j\} \rhd t \mid X_1 \rhd x_1, \ldots, X_{i-1} \rhd x_{i-1}, X_{i+1} \rhd x_{i+1}, \ldots, X_k \rhd x_k]}$$

$$(2) \quad \frac{[R \rhd t \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k] \quad x_i \notin \mathrm{var}(t)}{[R \rhd t \mid X_1 \rhd x_1, \ldots, X_{i-1} \rhd x_{i-1}, X_{i+1} \rhd x_{i+1}, \ldots, X_k \rhd x_k]}$$

Example 4 Consider the fact

$$\mathsf{f} = [dec(enc(X_1, X_2), X_3) \rhd x_1 \mid X_1 \rhd x_1, X_2 \rhd y, X_3 \rhd y].$$

We start by applying Rule (1), after which we obtain

$$[dec(enc(X_1, X_2), X_2) \rhd x_1 \mid X_1 \rhd x_1, X_2 \rhd y].$$

We continue with the application of Rule (2), after which we obtain the canonical form

$$\mathsf{f}' = [dec(enc(X_1, X_2), X_2) \rhd x_1 \mid X_1 \rhd x_1].$$

A *knowledge base* is a tuple (F, E) where F is a finite set of well-formed deduction facts that are in canonical form and E a finite set of equational facts.

Definition 5 (update) Given a fact $\mathsf{f} = [R \rhd t \mid X_1 \rhd t_1, \ldots, X_n \rhd t_n]$ and a knowledge base (F, E), *the update of* (F, E) *by* f, written $(\mathsf{F}, \mathsf{E}) \oplus \mathsf{f}$, is defined as

$$\begin{cases} (\mathsf{F} \cup \{\mathsf{f}'\}, \mathsf{E}) & \text{if } \mathsf{f} \text{ is solved and } \mathsf{F} \not\vdash t \qquad \textsf{useful fact} \\ \qquad \text{where } \mathsf{f}' \text{ is the canonical form of } \mathsf{f} & \\ (\mathsf{F}, \mathsf{E} \cup \{[R' \sim R\sigma \mid \emptyset]\}) & \text{if } \mathsf{f} \text{ is solved and } \mathsf{F} \vdash t \;\; \textsf{redundant fact} \\ \qquad \text{where } \mathsf{F} \vdash^{R'} t \text{ and } \sigma = \{X_1 \mapsto t_1, \ldots, X_n \mapsto t_n\} & \\ (\mathsf{F} \cup \{\mathsf{f}\}, \mathsf{E}) & \text{if } \mathsf{f} \text{ is not solved} \qquad \textsf{unsolved fact} \end{cases}$$

The choice of the recipe R' in the *redundant fact* case is defined by the implementation. While this choice does not influence the correctness of the procedure, it might influence its termination as we will see later. Note that, the result of updating a knowledge base by a (possibly not well-formed and/or not canonical) fact is again a knowledge base. Facts that are not well-formed will be captured by the *redundant fact* case, which adds an equational fact.

The role of the update function is to add facts to the knowledge base, while performing some redundancy elimination. If $\mathsf{F} \not\rhd t$, then the new fact clearly provides interesting information and it is added to the knowledge base. If the new fact is unsolved, it is added anyway (because it might prove useful later on). If the new fact is solved and $\mathsf{F} \rhd t$, then this deduction fact does not provide new information about deducible terms, but it might provide a new recipe for terms we already know deducible. Therefore, an equational fact is added instead, stating that the two recipes are equal provided the required side conditions are satisfied.

Example 5 We consider the knowledge base formed of the following set F of deduction facts:

$$\begin{array}{lcclr}
[\ w_1 & \rhd & enc(b,k) & \mid \emptyset] & (\mathsf{f}_1)\\
[\ b & \rhd & b & \mid \emptyset] & (\mathsf{f}_2)\\
[\ enc(Y_1,Y_2) & \rhd & enc(y_1,y_2) & \mid Y_1 \rhd y_1, Y_2 \rhd y_2] & (\mathsf{f}_3)
\end{array}$$

and the empty set E of equational facts.

We have already seen that $enc(enc(b,k),b)$ is generated by F with recipe $enc(w_1,b)$. Updating the knowledge base by $[w_2 \rhd enc(enc(b,k),b) \mid \emptyset]$ would result in no modification of the set of deduction facts, since we already know that $enc(enc(b,k),b)$ is generated. However, a new equational fact $[w_2 \sim enc(w_1,b) \mid \emptyset]$ would be added to the set of equational facts.

*Initialisation.*Given a frame $\varphi = \nu\tilde{n}.\{w_1 \mapsto t_1,\ldots,w_n \mapsto t_n\}$, our procedure starts from an *initial knowledge base* associated to φ and defined as follows:

$$\begin{array}{rl}
\mathrm{Init}(\varphi) = & (\emptyset,\emptyset)\\
\bigoplus_{1\le i\le n} & [w_i \rhd t_i]\\
\bigoplus_{n\in \mathrm{fn}(\varphi)} & [n \rhd n]\\
\bigoplus_{f\in\mathcal{F}} & [f(X_1,\ldots,X_k) \rhd f(x_1,\ldots,x_k) \mid X_1 \rhd x_1, \ldots \rhd X_k \rhd x_k]
\end{array}$$

Example 6 Consider the rewriting system $\mathcal{R}_{\mathcal{E}_{mal}}$ and $\varphi_2 = \nu a,k.\{w_1 \mapsto enc(b,k)\}$. The knowledge base $\mathrm{Init}(\varphi_2)$ is made up of the following deduction facts:

$$\begin{array}{lcclr}
[\ w_1 & \rhd & enc(b,k) & \mid \emptyset] & (\mathsf{f}_1)\\
[\ b & \rhd & b & \mid \emptyset] & (\mathsf{f}_2)\\
[\ enc(Y_1,Y_2) & \rhd & enc(y_1,y_2) & \mid Y_1 \rhd y_1, Y_2 \rhd y_2] & (\mathsf{f}_3)\\
[\ dec(Y_1,Y_2) & \rhd & dec(y_1,y_2) & \mid Y_1 \rhd y_1, Y_2 \rhd y_2] & (\mathsf{f}_4)\\
[\ mal(Y_1,Y_2) & \rhd & mal(y_1,y_2) & \mid Y_1 \rhd y_1, Y_2 \rhd y_2] & (\mathsf{f}_5)
\end{array}$$

*Saturation.*The aim of our saturation procedure is to produce

1. a set of solved deduction facts which have the same set of syntactic consequences as the initial set of deduction facts modulo the equational theory;

2. a set of solved equational facts whose consequences are exactly the equations holding in the frame.

The main part of this procedure consists in saturating the knowledge base $\mathrm{Init}(\varphi)$ by means of the transformation rules described in Figure 1. The rule Narrowing is designed to apply a rewriting step on an existing deduction fact. Intuitively, this rule allows us to get rid of the equational theory and nevertheless ensures that the generation of deducible terms is complete. This rule might introduce unsolved side conditions. The rule F-Solving is then used to instantiate the unsolved side conditions of an existing deduction fact. Unifying and E-Solving add equational facts which remember when different recipes for the same term exist.

Note that this procedure may not terminate and that the fixed point may not be unique (the $\oplus$ operation that adds a new fact to a knowledge base is not commutative).

We write $\Longrightarrow^*$ for the reflexive and transitive closure of $\Longrightarrow$.

Narrowing

$$\frac{\begin{array}{l}\mathsf{f} = [M \rhd C[t] \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k] \in \mathsf{F},\ l \to r \in \mathcal{R}_{\mathcal{E}} \\ \text{with } t \notin \mathcal{X},\ \ \sigma = \mathrm{mgu}(l, t)\ \text{ and }\ \mathrm{var}(\mathsf{f}) \cap \mathrm{var}(l) = \emptyset.\end{array}}{(\mathsf{F}, \mathsf{E}) \Longrightarrow (\mathsf{F}, \mathsf{E}) \oplus \mathsf{f}_0}$$

where $\mathsf{f}_0 = [M \rhd (C[r])\sigma \mid X_1 \rhd x_1\sigma, \ldots, X_k \rhd x_k\sigma]$.

F-Solving

$$\frac{\begin{array}{l}\mathsf{f}_1 = [M \rhd t \mid X \rhd u, X_1 \rhd t_1, \ldots, X_k \rhd t_k],\ \mathsf{f}_2 = [N \rhd s \mid Y_1 \rhd y_1, \ldots, Y_\ell \rhd y_\ell] \in \mathsf{F} \\ \text{with } u \notin \mathcal{X},\ \ \ \sigma = \mathrm{mgu}(s, u)\ \ \text{ and }\ \ \mathrm{var}(\mathsf{f}_1) \cap \mathrm{var}(\mathsf{f}_2) = \emptyset.\end{array}}{(\mathsf{F}, \mathsf{E}) \Longrightarrow (\mathsf{F}, \mathsf{E}) \oplus \mathsf{f}_0}$$

where $\mathsf{f}_0 = [M\{X \mapsto N\} \rhd t\sigma \mid \{X_i \rhd t_i\sigma\}_{1 \le i \le k} \cup \{Y_i \rhd y_i\sigma\}_{1 \le i \le \ell}]$.

Unifying

$$\frac{\begin{array}{l}\mathsf{f}_1 = [M \rhd t \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k],\ \mathsf{f}_2 = [N \rhd s \mid Y_1 \rhd y_1, \ldots, Y_\ell \rhd y_\ell] \in \mathsf{F} \\ \text{with } \sigma = \mathrm{mgu}(s, t)\ \ \text{ and }\ \ \mathrm{var}(\mathsf{f}_1) \cap \mathrm{var}(\mathsf{f}_2) = \emptyset.\end{array}}{(\mathsf{F}, \mathsf{E}) \Longrightarrow (\mathsf{F}, \mathsf{E} \cup \{\mathsf{f}_0\})}$$

where $\mathsf{f}_0 = [M \sim N \mid \{X_i \rhd x_i\sigma\}_{1 \le i \le k} \cup \{Y_i \rhd y_i\sigma\}_{1 \le i \le \ell}]$.

E-Solving

$$\frac{\begin{array}{l}\mathsf{f}_1 = [U \sim V \mid Y \rhd s, X_1 \rhd t_1, \ldots, X_k \rhd t_k] \in \mathsf{E},\ \mathsf{f}_2 = [M \rhd t \mid Y_1 \rhd y_1, \ldots, Y_\ell \rhd y_\ell\} \in \mathsf{F} \\ \text{with } s \notin \mathcal{X},\ \sigma = \mathrm{mgu}(s, t) \text{ and } \mathrm{var}(\mathsf{f}_1) \cap \mathrm{var}(\mathsf{f}_2) = \emptyset.\end{array}}{(\mathsf{F}, \mathsf{E}) \Longrightarrow (\mathsf{F}, \mathsf{E} \cup \{\mathsf{f}_0\})}$$

where $\mathsf{f}_0 = [U\{Y \mapsto M\} \sim V\{Y \mapsto M\} \mid \{X_i \rhd t_i\sigma\}_{1 \le i \le k} \cup \{Y_i \rhd y_i\sigma\}_{1 \le i \le \ell}]$.

Fig. 1 Saturation rules

Example 7 Continuing Example 6, we illustrate the saturation procedure. We can apply the rule Narrowing on fact f_4 and rewrite rule $dec(enc(x, y), y) \to x$, as well as on fact f_5 and rewrite rule $mal(enc(x, y), z) \to enc(z, y)$ adding facts

$$\begin{array}{lcll} [dec(Y_1, Y_2) \rhd & x & \mid Y_1 \rhd enc(x, y), Y_2 \rhd y] & \qquad (\mathsf{f}_6) \\ [mal(Y_1, Y_2) \rhd & enc(z, y) & \mid Y_1 \rhd enc(x, y), Y_2 \rhd z] & \qquad (\mathsf{f}_7) \end{array}$$

The facts f_6 and f_7 are not solved and we can apply the rule F-Solving with f_1 adding the facts:

$$\begin{array}{llll} [dec(w_1, Y_2) \rhd & b & \mid Y_2 \rhd k] & (\mathsf{f}_8) \\ [mal(w_1, Y_2) \rhd & enc(z, k) & \mid Y_2 \rhd z] & (\mathsf{f}_9) \end{array}$$

Rule Unifying can be used on facts $\mathsf{f}_1/\mathsf{f}_3$, $\mathsf{f}_3/\mathsf{f}_9$ as well as $\mathsf{f}_1/\mathsf{f}_9$ to add equational facts. This third case allows one to obtain $\mathsf{f}_{10} = [w_1 \sim mal(w_1, Y_2) \mid Y_2 \rhd b]$ which can be solved (using E-Solving with f_2) to obtain $\mathsf{f}_{11} = [w_1 \sim mal(w_1, b)]$, etc. When reaching a fixed point, f_9, f_{11} and the facts in $\mathrm{Init}(\varphi_2)$ are some of the solved facts contained in the knowledge base.

We now state the soundness and completeness of our transformation rules. The technical lemmas used to prove this result are detailed in Section 4 (see also Appendix A).

Theorem 1 (soundness and completeness) *Let φ be a frame and (F, E) be a saturated knowledge base such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$. Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{N})$ and $\mathsf{F}^+ = \mathsf{F} \cup \{[n \rhd n] \mid n \in \mathrm{fn}(t) \smallsetminus \mathrm{bn}(\varphi)\}$. We have that:*

1. *For all $M \in \mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathrm{dom}(\varphi))$ such that $\mathrm{fn}(M) \cap \mathrm{bn}(\varphi) = \emptyset$, we have that*

$$M\varphi =_{\mathcal{E}} t \Leftrightarrow \exists N,\ \mathsf{E} \models M \sim N \text{ and } \mathsf{F}^+ \vdash^N t{\downarrow_{\mathcal{R}_\mathcal{E}}}$$

2. *For all $M, N \in \mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathrm{dom}(\varphi))$ such that $\mathrm{fn}(M, N) \cap \mathrm{bn}(\varphi) = \emptyset$, we have*

$$(M =_{\mathcal{E}} N)\varphi \ \Leftrightarrow\ \mathsf{E} \models M \sim N.$$

3.3 Application to deduction and static equivalence

Procedure for deduction. Let φ be a frame and t be a ground term. The procedure for checking $\varphi \vdash_{\mathcal{E}} t$ runs as follows:

1. Apply the saturation rules to obtain (if any) a saturated knowledge base (F, E) such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$. Let $\mathsf{F}^+ = \mathsf{F} \cup \{[n \rhd n] \mid n \in \mathrm{fn}(t) \smallsetminus \mathrm{bn}(\varphi)\}$.
2. Return *yes* if there exists N such that $\mathsf{F}^+ \vdash^N t{\downarrow_{\mathcal{R}_\mathcal{E}}}$ (that is, the $\mathcal{R}_\mathcal{E}$-normal form of t is generated by F with recipe N); otherwise return *no*.

Proof If the algorithm returns *yes*, there exists N such that $\mathsf{F}^+ \vdash^N t{\downarrow_{\mathcal{R}_\mathcal{E}}}$. As $\mathsf{E} \models N \sim N$, by Theorem 1 we have that $N\varphi =_{\mathcal{E}} t{\downarrow_{\mathcal{R}_\mathcal{E}}}$, *i.e.*, $\varphi \vdash_{\mathcal{E}} t$. Conversely, if t is deducible from φ, then there exists M such that $M\varphi =_{\mathcal{E}} t$. By Theorem 1, there exists N such that $\mathsf{F}^+ \vdash^N t{\downarrow_{\mathcal{R}_\mathcal{E}}}$. Hence, the algorithm returns *yes*. □

Example 8 We continue our running example. Let (F, E) be the knowledge base obtained from $\mathrm{Init}(\varphi_2)$ described in Example 7. We show that $\varphi_2 \vdash enc(c, k)$ and $\varphi_2 \vdash b$. Indeed we have that $\mathsf{F} \cup \{[c \rhd c]\} \vdash^{mal(w_1, c)} enc(c, k)$ using facts f_9 and $[c \rhd c]$, and $\mathsf{F} \vdash^b b$ using fact f_2.

Procedure for static equivalence. Let φ_1 and φ_2 be two frames. The procedure for checking $\varphi_1 \approx_{\mathcal{E}} \varphi_2$ runs as follows:

1. Apply the transformation rules to obtain (if possible) two saturated knowledge bases $(\mathsf{F}_i, \mathsf{E}_i)$, $i = 1, 2$ such that $\mathrm{Init}(\varphi_i) \Longrightarrow^* (\mathsf{F}_i, \mathsf{E}_i)$, $i = 1, 2$.

2. For $\{i,j\} = \{1,2\}$, for every solved fact $[M \sim N \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k]$ in E_i, check if $(M\sigma =_{\mathcal{E}} N\sigma)\varphi_j$ where $\sigma = \{X_1 \mapsto x_1, \ldots, X_k \mapsto x_k\}$.
3. If so return *yes*; otherwise return *no*.

Proof If the algorithm returns *yes*, this means that ($\star$): for every solved equational fact $[M \sim N \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k]$ in E_1, we have that:

$$(M\sigma =_{\mathcal{E}} N\sigma)\varphi_2$$

where $\sigma = \{X_1 \mapsto x_1, \ldots, X_k \mapsto x_k\}$. Let $M, N \in \mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathrm{dom}(\varphi))$ such that $\mathrm{fn}(M,N) \cap \tilde{n} = \emptyset$ and $(M =_{\mathcal{E}} N)\varphi_1$. Thanks to Theorem 1, we have that $\mathsf{E}_1 \models M \sim N$. Thanks to ($\star$), we deduce that $(M =_{\mathcal{E}} N)\varphi_2$. The other direction is proved in the same way.

Conversely, assume now that $\varphi_1 \approx_{\mathcal{E}} \varphi_2$. Let $[M \sim N \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k]$ be a solved equational fact in E_1 and let us show that $(\tilde{M} =_{\mathcal{E}} \tilde{N})\varphi_2$ where

– $\tilde{M} = M\{X_1 \mapsto x_1, \ldots, X_k \mapsto x_k\}$, and
– $\tilde{N} = N\{X_1 \mapsto x_1, \ldots, X_k \mapsto x_k\}$.

(The other case is done in a similar way, and we will conclude that the algorithm returns *yes*.) Let $\{y_1, \ldots, y_\ell\} = \mathrm{var}(M,N)$ and $n_1, \ldots, n_\ell$ be ℓ fresh names that occur neither in $\tilde{n} \cup \mathrm{fn}(M,N)$, nor in φ. Let $\delta = \{y_1 \mapsto n_1, \ldots, y_\ell \mapsto n_\ell\}$. Since $\mathsf{E}_1 \models \tilde{M} \sim \tilde{N}$, we have also that $\mathsf{E}_1 \models \tilde{M}\delta \sim \tilde{N}\delta$. Clearly, we have that $\mathrm{fn}(\tilde{M}\delta, \tilde{N}\delta) \cap \tilde{n} = \emptyset$, thus by Theorem 1, we have that $(\tilde{M}\delta =_{\mathcal{E}} \tilde{N}\delta)\varphi_1$. As $\varphi_1 \approx_{\mathcal{E}} \varphi_2$, we have also that $(\tilde{M}\delta =_{\mathcal{E}} \tilde{N}\delta)\varphi_2$, and thus $(\tilde{M} =_{\mathcal{E}} \tilde{N})\varphi_2$. This allows us to conclude. $\square$

Example 9 Consider again the frames φ_1 and φ_2 which are not statically equivalent (see Example 2). Our procedure answers *no* since $[\mathit{mal}(w_1, b) \sim w_1] \in \mathsf{E}_2$ whereas $(\mathit{mal}(w_1, b) \neq_{\mathcal{E}_{mal}} w_1)\varphi_1$.

4 Soundness and completeness

In this section we give the key results which are used to prove the two directions of Theorem 1.

We now define when a fact makes a valid statement about a given frame φ. We say that the fact *holds* in φ.

Definition 6 (f holds in φ) Let φ be a frame and $\mathsf{f} = [R \rhd t \mid \Delta]$ (respectively $[U \sim V \mid \Delta]$) be a fact with $\Delta = \{X_1 \rhd t_1, \ldots, X_k \rhd t_k\}$. We say that f *holds* in φ if for any substitution τ grounding for $t_1, \ldots, t_k$ such that $\varphi \vdash_{\mathcal{E}} t_i\tau$ with recipe R_i for $1 \leq i \leq n$, we have that $\varphi \vdash_{\mathcal{E}} t\tau$ with recipe $R\{X_i \mapsto R_i\}_{1 \leq i \leq k}$ (respectively $(U\{X_i \mapsto R_i\}_{1 \leq i \leq k} =_{\mathcal{E}} V\{X_i \mapsto R_i\}_{1 \leq i \leq k})\varphi$).

Example 10 Consider the fact $\mathsf{f}_9 = [\mathit{mal}(w_1, Y_2) \rhd \mathit{enc}(z, k) \mid Y_2 \rhd z]$ and the frame $\varphi_2 = \nu a, k.\{w_1 \mapsto \mathit{enc}(b, k)\}$ given in Example 7. We have that f_9 holds in φ_2. Indeed, supposing t_1 is a term such that $\varphi_2 \vdash_{\mathcal{E}} t_1$ with recipe R_1, we have that $\varphi_2 \vdash_{\mathcal{E}} \mathit{enc}(t_1, k)$ with recipe $\mathit{mal}(w_1, R_1)$: $\mathit{mal}(w_1, R_1)\varphi_2 = \mathit{mal}(\mathit{enc}(b, k), t_1) = \mathit{enc}(t_1, k)$.

4.1 Soundness

Lemma 1 ensures that any knowledge base obtained from $\mathrm{Init}(\varphi)$ will only contain facts that hold in φ.

Lemma 1 *Let* φ *be a frame and* (F, E) *be a knowledge base such that* $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$. *Then every* $\mathsf{f} \in \mathsf{F} \cup \mathsf{E}$ *holds in* φ.

Intuitively Lemma 2 states that any ground term which can be generated is indeed deducible. Similarly all equations which are consequences of the knowledge base are true equations in the initial frame. The soundness of our saturation procedure can be easily derived from this lemma.

Lemma 2 (soundness) *Let* φ *be a frame and* (F, E) *be a knowledge base such that* $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$. *Let* $t \in \mathcal{T}(\mathcal{F}, \mathcal{N})$, $M, N \in \mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathrm{dom}(\varphi))$ *be a term such that* $\mathrm{fn}(M, N) \cap \mathrm{bn}(\varphi) = \emptyset$, *and* $\mathsf{F}^+ = \mathsf{F} \cup \{[n \rhd n] \mid n \in \mathrm{fn}(t) \smallsetminus \mathrm{bn}(\varphi)\}$. *We have that:*

1. $\mathsf{F}^+ \vdash^M t \;\Rightarrow\; M\varphi =_{\mathcal{E}} t$; *and*
2. $\mathsf{E} \models M \sim N \;\Rightarrow\; (M =_{\mathcal{E}} N)\varphi$.

Proof By Lemma 1 and because every $\mathsf{f} \in \{[n \rhd n] \mid n \in \mathrm{fn}(t) \smallsetminus \mathrm{bn}(\varphi)\}$ holds in φ, we have that all facts in F^+ hold in φ. To conclude, we show Points 1 and 2 stated in the Lemma.

1. Let M and t be such that $\mathsf{F}^+ \vdash^M t$. By definition of $\vdash$, as t is ground, there exists a solved deduction fact $\mathsf{f}_0 = [M_0 \rhd t_0 \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k] \in \mathsf{F}^+$ such that $t = t_0\sigma$ for some substitution σ and $\mathsf{F}^+ \vdash^{M_i} x_i\sigma$ for some M_i $(1 \leq i \leq k)$ and $M = M_0\{X_1 \mapsto M_1, \ldots, X_k \mapsto M_k\}$. We show the result by induction on $|t|$.
   Base case: $|t| = 1$. In such a case t is either a name or a constant. We have that $k = 0$, $t_0 = t$ and $M = M_0$. Since f_0 holds in φ, we deduce that $\varphi \vdash_{\mathcal{E}} t$ with recipe M_0, i.e. $M_0\varphi =_{\mathcal{E}} t$. This allows us to conclude.

   Induction step. Note that $|x_i\sigma| < |t|$ and $\mathsf{F}^+ \vdash^{M_i} x_i\sigma$, thus we can apply our induction hypothesis on $x_i\sigma$. We deduce that $M_i\varphi =_{\mathcal{E}} x_i\sigma$ and thus $M\varphi =_{\mathcal{E}} t_0\sigma = t$ since f_0 holds in φ.
2. Let M and N be such that $\mathrm{fn}(M, N) \cap \mathrm{bn}(\varphi) = \emptyset$ and $\mathsf{E} \models M \sim N$. To show that $(M =_{\mathcal{E}} N)\varphi$, it is sufficient to establish that
   $$(M'\sigma =_{\mathcal{E}} N'\sigma)\varphi \quad \text{where } \sigma = \{X_1 \mapsto x_1, \ldots, X_k \mapsto x_k\}$$
   for every solved equational fact $[M' \sim N' \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k] \in \mathsf{E}$. This follows easily from Lemma 1. □

4.2 Completeness

We now give two propositions that are used to show the completeness of the saturation rules. The first one states that whenever there exist two recipes to generate a ground term from F then the equation on the two recipes is a consequence of E.

Lemma 3 *Let* (F, E) *be a saturated knowledge base and* $\mathsf{f} = [U \sim V \mid X_1 \rhd t_1, \ldots, X_k \rhd t_k]$ *be an equational fact in* E. *For any substitution* σ *grounding for* $\{t_1, \ldots, t_k\}$ *such that* $\mathsf{F} \vdash t_i\sigma$ $(1 \leq i \leq k)$, *we have that* $\mathsf{F} \vdash^{R_i} t_i\sigma$ *for some* R_i $(1 \leq i \leq k)$ *and* $\mathsf{E} \models U\tau \sim V\tau$ *where* $\tau = \{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}$.

Proposition 1 (completeness, equation) *Let* (F, E) *be a saturated knowledge base, and* M, N *be two terms such that* $\mathsf{F} \vdash^{M} t$ *and* $\mathsf{F} \vdash^{N} t$ *for some ground term* t. *Then, we have that* $\mathsf{E} \models M \sim N$.

Proof By definition of $\mathsf{F} \vdash^M t$ we know that there exist a substitution σ_1 and a deduction fact $\mathsf{f}_1 = [M_0 \rhd u_0 \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k]$ in F such that $u_0\sigma_1 = t$, $\mathsf{F} \vdash^{M_i} x_i\sigma_1$ $(1 \leq i \leq k)$ and $M_0\{X_i \mapsto M_i\}_{1 \leq i \leq k} = M$. Similarly, by definition of $\mathsf{F} \vdash^N t$ we know that there exist a substitution σ_2 and a deduction fact $\mathsf{f}_2 = [N_0 \rhd v_0 \mid Y_1 \rhd y_1, \ldots, Y_\ell \rhd y_\ell]$ in F such that $v_0\sigma_2 = t$, $\mathsf{F} \vdash^{N_j} y_j\sigma_2$ $(1 \leq j \leq \ell)$ and $N_0\{Y_j \mapsto N_j\}_{1 \leq i \leq \ell} = N$.

We prove the result by induction on $|t|$. As our knowledge base (F, E) is saturated, rule $\mathsf{Unifying}$ must have been applied to the facts f_1 and f_2. Therefore, we have that there exists an equational fact $\mathsf{f}_3 \in \mathsf{E}$ such that:

$$\mathsf{f}_3 = [M_0 \sim N_0 \mid X_1 \rhd x_1\sigma, \ldots, X_k \rhd x_k\sigma, Y_1 \rhd y_1\sigma, \ldots, Y_\ell \rhd y_\ell\sigma].$$

where $\sigma = mgu(u_0, v_0)$.

Let σ' be a substitution such that $\sigma_1 \cup \sigma_2 = \sigma \circ \sigma'$. We can now apply Lemma 3 on f_3 with substitution σ'. We obtain that there exist $R_1, \ldots, R_k$ and $W_1, \ldots, W_\ell$ such that $\mathsf{F} \vdash^{R_i} x_i\sigma\sigma'$ $(1 \leq i \leq k)$ and $\mathsf{F} \vdash^{W_j} y_j\sigma\sigma'$ $(1 \leq j \leq \ell)$ and such that

$$\mathsf{E} \models M_0\delta \sim N_0\delta \qquad (1)$$

where $\delta = \{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k, Y_1 \mapsto W_1, \ldots, Y_\ell \mapsto W_\ell\}$.

As M_i and R_i $(1 \leq i \leq k)$ are such that $\mathsf{F} \vdash^{M_i} x_i\sigma_1$ and $\mathsf{F} \vdash^{R_i} x_i\sigma\sigma'$, and as $x_1\sigma\sigma' = x_1\sigma_1$ is a strict subterm of $u_0\sigma_1 = t$, we can apply the induction hypothesis to obtain that $\mathsf{E} \models M_i \sim R_i$. In a similar way, we also deduce that $\mathsf{E} \models N_j \sim W_j$ $(1 \leq j \leq \ell)$. By replacing W_j by M_j and R_i by N_i in equation (1), we obtain our conclusion. □

Next we show that whenever a ground term (not necessarily in normal form) can be generated then its normal form can also be generated and there exists an equation on the two recipes. This is the purpose of Proposition 2.

Lemma 4 *Let* (F, E) *be a saturated knowledge base. Let* $\mathsf{f} = [R \rhd t \mid X_1 \rhd t_1, \ldots, X_k \rhd t_k]$ *be a deduction fact such that* $(\mathsf{F}, \mathsf{E}) \oplus \mathsf{f} = (\mathsf{F}, \mathsf{E})$. *For any substitution* σ *grounding for* $\{t_1, \ldots, t_k\}$ *such that* $\mathsf{F} \vdash t_i\sigma$ $(1 \leq i \leq k)$, *we have that there exist* $R_1, \ldots, R_k$ *and* W *such that*

- $\mathsf{F} \vdash^{W} t\sigma$, *and* $\mathsf{F} \vdash^{R_i} t_i\sigma$ *for* $1 \leq i \leq k$;
- $\mathsf{E} \models W \sim R\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}$.

Proposition 2 (completeness, reduction) *Let* (F, E) *be a saturated knowledge base,* M *a term and* t *a ground term such that* $\mathsf{F} \vdash^M t$ *and* $t{\downarrow_{\mathcal{R}_\mathcal{E}}} \neq t$. *Then there exist* M' *and* t' *such that* $\mathsf{F} \vdash^{M'} t'$ *with* $t \rightarrow^{+}_{\mathcal{R}_\mathcal{E}} t'$ *and* $\mathsf{E} \models M \sim M'$.

Proof We show this result by induction on $|t|$. By definition of $\mathsf{F} \vdash^M t$ we know that there exist $\mathsf{f}_0 = \{M_0 \rhd u_0 \mid X_1 \rhd x_1, \dots, X_k \rhd x_k\}$ in F and a substitution σ such that $u_0\sigma = t$ and $\mathsf{F} \vdash^{M_i} x_i\sigma$ $(1 \leq i \leq k)$ and $M_0\{X_i \mapsto M_i\}_{1 \leq i \leq k} = M$ for some M_i $(1 \leq i \leq k)$. We distinguish two cases:

Case 1: there exists $1 \leq j \leq k$ *such that* $x_j\sigma{\downarrow_{\mathcal{R}_\mathcal{E}}} \neq x_j\sigma$. Let us assume w.l.o.g. that $j = 1$. Since $x_1\sigma$ is a strict subterm of t, we can apply our induction hypothesis on $x_1\sigma$. We obtain that there exist M_1' and u_1' such that $\mathsf{F} \vdash^{M_1'} u_1'$ with $x_1\sigma \rightarrow^+_{\mathcal{R}} u_1'$ and $\mathsf{E} \models M_1 \sim M_1'$. Now, let σ' be the substitution defined as follows:

$$x\sigma' = \begin{cases} x\sigma \text{ for } x \neq x_1 \\ u_1' \text{ otherwise} \end{cases}$$

Let $t' = u_0\sigma'$ and $M' = M_0\{X_1 \mapsto M_1', X_2 \mapsto M_2, \dots, X_k \mapsto M_k\}$. Since $x_1 \in \mathrm{var}(u_0)$, it is easy to see that $t = u_0\sigma \rightarrow^+_{\mathcal{R}} u_0\sigma' = t'$. Furthermore, it is also easy to see that $\mathsf{F} \vdash^{M'} t'$. Lastly, since $\mathsf{E} \models M_1 \sim M_1'$, we have that $\mathsf{E} \models M \sim M'$.

Case 2: $x_j\sigma{\downarrow_{\mathcal{R}_\mathcal{E}}} = x_j\sigma$ *for every* $1 \leq j \leq k$. In such a case, we have that $u_0 = C[u_0']$ for some context C and some term $u_0' \notin \mathcal{X}$ such that $u_0'\sigma = l\tau$ where $l \rightarrow r \in \mathcal{R}$ and τ is a substitution. As the knowledge base (F, E) is saturated, the rule $\mathsf{Narrowing}$ must have been applied. Therefore there exists f_1 such that:

– $(\mathsf{F}, \mathsf{E}) \oplus \mathsf{f}_1 = (\mathsf{F}, \mathsf{E})$, and
– $\mathsf{f}_1 = [M_0 \rhd (C[r])\rho \mid X_1 \rhd x_1\rho, \dots, X_k \rhd x_k\rho]$

where $\rho = \mathrm{mgu}(u_0', l)$. Let ρ' be the substitution with $\mathrm{dom}(\rho') = \mathrm{var}(\{x_1\rho, \dots, x_k\rho\})$ and $\sigma \cup \tau = \rho \circ \rho'$. Now, we apply Lemma 4 on the fact f_1 and the substitution ρ'. We deduce that there exist $R_1, \dots, R_k$ and W such that

– $\mathsf{F} \vdash^W (C[r])\rho\rho'$, and $\mathsf{F} \vdash^{R_i} x_i\rho\rho'$ for $1 \leq i \leq k$; and
– $\mathsf{E} \models W \sim M_0\{X_1 \mapsto R_1, \dots, X_k \mapsto R_k\}$.

Let $t' = (C[r])\rho\rho'$ and $M' = W$. We have that $\mathsf{F} \vdash^{M'} t'$. Moreover, since $\mathsf{F} \vdash^{R_i} x_i\rho\rho'$, $\mathsf{F} \vdash^{M_i} x_i\sigma$ and $x_i\rho\rho' = x_i\sigma$, we can apply Lemma 1 in order to deduce that $\mathsf{E} \models R_1 \sim M_i$ for $1 \leq i \leq k$. Thus, we have that $\mathsf{E} \models M \sim M'$. In order to conclude, it remains to show that $t \rightarrow^+_{\mathcal{R}_\mathcal{E}} t'$. Indeed, we have that $t = u_0\sigma = (C[u_0'])\sigma \rightarrow^+_{\mathcal{R}_\mathcal{E}} (C[r])\rho\rho' = t'$. □

Relying on these propositions, we can show completeness of our saturation procedure (i.e. $\Rightarrow$ of Theorem 1).

1. To prove Item 1, we first observe that if t is deducible from φ modulo $\mathcal{E}$ then $\mathsf{F}^+ \vdash^{M'} t_0$ for some M' and t_0 such that $\mathsf{E} \models M \sim M'$ and $t_0 \rightarrow^* t{\downarrow_{\mathcal{R}_\mathcal{E}}}$. Actually M' differs from M by the fact that some public names that do not occur in the knowledge base are replaced by fresh variables. Then, we rely on Proposition 2 and we show the result by induction on t_0 equipped with the order $<$ induced by the rewriting relation ($t < t'$ iff $t \rightarrow^+ t'$).
2. Now, to prove Item 2, we apply the result shown in Item 1 on $M\varphi =_\mathcal{E} t$ and $N\varphi =_\mathcal{E} t$ where $t = M\varphi{\downarrow_{\mathcal{R}_\mathcal{E}}} = N\varphi{\downarrow_{\mathcal{R}_\mathcal{E}}}$. We deduce that there exist M' and N' such that $\mathsf{E} \models M \sim M'$, $\mathsf{F}^+ \vdash^{M'} t$, $\mathsf{E} \models N \sim N'$, and $\mathsf{F}^+ \vdash^{N'} t$. Then, Proposition 1 allows one to deduce that $\mathsf{E} \models M' \sim N'$, thus $\mathsf{E} \models M \sim N$.

5 Termination

As already announced the saturation process will not always terminate.

Example 11 Consider the convergent rewriting system consisting of the single rule $f(g(x)) \to g(h(x))$ and the frame $\phi = \nu a.\{w_1 \mapsto g(a)\}$. We have that

$$\mathrm{Init}(\varphi) \supseteq \{[w_1 \rhd g(a)],\ [f(X) \rhd f(x) \mid X \rhd x]\}.$$

By $\mathsf{Narrowing}$ we can add the fact $\mathsf{f}_1 = [f(X) \rhd g(h(x)) \mid X \rhd g(x)]$. Then we can apply $\mathsf{F\text{-}Solving}$ to solve its side condition $X \rhd g(x)$ with the fact $[w_1 \rhd g(a)]$ yielding the solved fact $[f(w_1) \rhd g(h(a))]$. Now, applying iteratively $\mathsf{F\text{-}Solving}$ on f_1 and the newly generated fact, we generate an infinity of solved facts of the form $[f(\ldots f(w_1)\ldots) \rhd g(h(\ldots h(a)\ldots))]$. Intuitively, this happens because our symbolic representation is unable to express that the function h can be nested an unbounded number of times when it occurs under an application of g.

The same kind of limitation already exists in the procedure implemented in the tool YAPA [10]. However, our symbolic representation which manipulates terms that are not necessarily ground and facts with side conditions allows us to go beyond YAPA. We are able for instance to treat equational theories such as malleable encryption and trapdoor commitment.

5.1 Generic method for proving termination

We provide a generic method for proving termination, which we instantiate in the following section on several examples.

In order to prove that the saturation algorithm terminates, we require that the update function $\oplus$ be *uniform*: i.e., the same recipe R' be used for all redundant solved deduction facts that have the same canonical form. Note that the soundness and completeness of the algorithm does not depend on the choice of the recipe R' when updating the knowledge base with a redundant fact (*cf.* Definition 5).

Definition 7 (projection) We define the *projection* of a deduction fact $\mathsf{f} = [R \rhd t \mid X_1 \rhd t_1, \ldots, X_n \rhd t_n]$ as $\hat{\mathsf{f}} = [t \mid \{t_1, \ldots, t_n\}]$. We extend the projection to sets of facts F and define $\hat{\mathsf{F}} = \{\hat{\mathsf{f}} \mid \mathsf{f} \in \mathsf{F}\}$.

We identify projections which are equal up to bijective renaming of variables and we sometimes omit braces for the side conditions.

Proposition 3 (generic termination) *The saturation algorithm terminates if* $\oplus$ *is uniform and there exist some functions* $\mathcal{Q}$, $\mathsf{m_f}$, $\mathsf{m_e}$ *and some well-founded orders* $<_\mathsf{f}$ *and* $<_\mathsf{e}$ *such that for all frames* φ, *and for all* (F, E) *such that* $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$, *we have that:*

1. $\{\hat{\mathsf{f}} \mid \mathsf{f} \in \mathsf{F}$ *and* f *is a solved deduction fact*$\} \subseteq \mathcal{Q}(\varphi)$ *and* $\mathcal{Q}(\varphi)$ *is finite;*
2. $\mathsf{m_f}(\mathsf{f}_0) <_\mathsf{f} \mathsf{m_f}(\mathsf{f}_1)$ *where* f_0, f_1 *are defined as in rule F-Solving;*
3. $\mathsf{m_e}(\mathsf{f}_0) <_\mathsf{e} \mathsf{m_e}(\mathsf{f}_1)$ *where* f_0, f_1 *are defined as in rule E-Solving.*

Proof A *solved deduction fact* f is only added to F if there is no $\mathsf{f}' \in \mathsf{F}$ such that $\hat{\mathsf{f}} = \hat{\mathsf{f}}'$. Indeed, if $\hat{\mathsf{f}} = \hat{\mathsf{f}}'$ then $\hat{\mathsf{f}}$ is redundant and an equational fact will be added instead. As $\{\hat{\mathsf{f}} \mid \mathsf{f} \in \mathsf{F} \text{ and } \mathsf{f} \text{ is a solved deduction fact}\} \subseteq \mathcal{Q}(\varphi)$ and $\mathcal{Q}(\varphi)$ is finite we conclude that only a finite number of solved deduction facts can be added.

An *unsolved deduction fact* f can be added in two ways.

- f can be added by the rule Narrowing. Since the number of solved deduction facts and the number of rewriting rules are finite the number of facts added by the rule Narrowing is bounded.
- f can be added by the rule F-Solving. The number of facts added by the rule F-Solving is bounded by the measure $\mathsf{m_f}$ which is strictly decreasing for a well-founded order.

An *equational fact* f can be added in three ways.

- f can be added when the knowledge base is updated with a redundant deduction fact. However, since $\oplus$ is uniform only a finite number of such facts is added.
- f can be added by the rule Unifying. Since the number of solved deduction facts is finite, the number of facts added by Unifying is bounded.
- f can be added by the rule E-Solving. The number of facts added by rule E-Solving is bounded by the measure $\mathsf{m_e}$ which is strictly decreasing for a well-founded order.

Altogether, this allows us to conclude. □

5.2 Applications

We now give several examples for which the saturation procedure indeed terminates. For each of these theories the definition of the function $\mathcal{Q}$ relies on the following notion of *extended subterm*.

Definition 8 (extended subterm) Let t be a term, its set of *extended subterms* $\mathrm{st}_{\mathcal{R}_\mathcal{E}}(t)$ (w.r.t. $\mathcal{E}$), is the smallest set such that:

1. $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(t)$,
2. $f(t_1, \ldots, t_k) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(t)$ implies $t_1, \ldots, t_k \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(t)$,
3. $t' \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(t)$ and $t' \to_{\mathcal{R}_\mathcal{E}} t''$ implies $t'' \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(t)$.

This notation is extended to frames in the usual way.

All examples in this section rely on the same $\mathsf{m_f}$ and $\mathsf{m_e}$. Let $\{X_1 \rhd t_1, \ldots, X_n \rhd t_n\}$ be the set of side conditions of a fact f. We define

$$\mathsf{m_f}(\mathsf{f}) = (\#\,\mathrm{var}(t_1, \ldots, t_n), \sum_{1 \leq i \leq n} |t_i|)$$

and $<_\mathsf{f}$ is the lexicographical order on ordered pairs of integers. The measure $\mathsf{m_e}$ and the order $<_\mathsf{e}$ are defined in the same way.

We now present the class of subterm convergent equational theories as well as the theories for malleable encryption and trap-door commitment. The detailed proofs are given in Appendix B.

5.2.1 Subterm convergent equational theories.

Abadi and Cortier [1] have shown that deduction and static equivalence are decidable for *subterm convergent* equational theories in polynomial time. We retrieve the same results with our algorithm. An equational theory $\mathcal{E}$ is subterm convergent if $\mathcal{R}_\mathcal{E}$ is convergent and for every rule $l \to r \in \mathcal{R}_\mathcal{E}$, we have that either r is a strict subterm of l, or r is a ground term in $\mathcal{R}_\mathcal{E}$-normal form.

The termination proof for this class relies on the function $\mathcal{Q}$ where $\mathcal{Q}(\varphi)$ is defined as the smallest set that contains

1. $[t \mid \emptyset]$, where $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$;
2. $[f(x_1, \ldots, x_k) \mid x_1, \ldots, x_k]$, where $ar(f) = k$.

5.2.2 Malleable encryption.

We also obtain termination for the equational theory $\mathcal{E}_{mal}$ described in Example 1. This is a toy example that does not fall in the class studied in [1]. Indeed, this theory is not *locally stable*: the set of terms in normal form deducible from a frame φ cannot always be obtained by applying public contexts over a finite set (called $sat(\varphi)$ in [1]) of ground terms.

As a witness consider the frame $\varphi_2 = \nu a, k.\{w_1 \mapsto enc(b, k)\}$ introduced in Example 2. Among the terms that are deducible from φ_2, we have those of the form $enc(t, k)$ where t represents any term deducible from φ_2. From this observation, it is easy to see that $\mathcal{E}_{mal}$ is not locally stable.

Our procedure does not have this limitation. A prerequisite for termination is that the set of terms in normal form deducible from a frame is exactly the set of terms obtained by nesting in all possible ways a finite set of contexts. The theory $\mathcal{E}_{mal}$ falls in this class. In particular, for the frame φ_2, our procedure produces the fact $\mathsf{f}_9 = [mal(w_1, Y_2) \rhd enc(z, k) \mid Y_2 \rhd z]$ allowing us to capture all the terms of the form $enc(t, k)$ by the means of a single deduction fact.

The termination proof relies on the function $\mathcal{Q}$ where $\mathcal{Q}(\varphi)$ is defined as the smallest set that contains:

1. $[t \mid \emptyset]$, for every $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$;
2. $[f(x_1, x_2) \mid x_1, x_2]$, where $f \in \{enc, dec, mal\}$;
3. $[enc(x, t) \mid x]$, if there exists t' such that $enc(t', t) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$.

5.2.3 Trap-door commitment.

The following convergent equational theory $\mathcal{E}_{td}$ is a model for trap-door commitment:

$$\begin{array}{ll} open(td(x,y,z),y) = x & td(x_2, f(x_1,y,z,x_2), z) = td(x_1,y,z) \\ open(td(x_1,y,z), f(x_1,y,z,x_2)) = x_2 & f(x_2, f(x_1,y,z,x_2), z, x_3) = f(x_1,y,z,x_3) \end{array}$$

As said in the introduction, we encountered this equational theory when studying electronic voting protocols. The term $td(m, r, td)$ models the commitment of the message m under the key r using an additional trap-door td. Such a commitment scheme allows a voter who has performed a commitment to open it in different ways using its trap-door. Hence, trap-door bit commitment $td(v, r, td)$ does not bind the voter to the vote v. This is useful to ensure privacy-type properties in e-voting and in particular receipt-freeness [25]. With such a scheme, even if a coercer requires the voter to reveal his commitment, this does not give any useful information to the coercer as the commitment can be viewed as the commitment of any vote (depending on the key that will be used to open it).

For the same reason as $\mathcal{E}_{mal}$, the theory of trap-door commitment described below cannot be handled by the algorithms described in [1,10]. Our termination proof relies on the function $\mathcal{Q}$ where $\mathcal{Q}(\varphi)$ is the smallest set that contains:

1. $[t \mid \emptyset]$, for every $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$;
2. $[td(t_1, r, tp) \mid \emptyset]$ such that $f(t_1, r, tp, t_2) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for some t_2;
3. $[g(x_1, \ldots, x_k) \mid x_1, \ldots, x_k]$, where $g \in \{open, td, f\}$ and $ar(g) = k$;
4. $[f(t_1, r, tp, x) \mid x]$, such that $f(t_1, r, tp, t_2) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for some t_2.

5.2.4 Blind signatures

The following convergent equational theory $\mathcal{E}_{blind}$ has been introduced in [22] for modeling blind signatures in e-voting protocols. Abadi and Cortier have shown that deduction and static equivalence are decidable for this theory [1].

1. $unblind(blind(x, y), y) = x$
2. $unblind(sign(blind(x, y), z), y) = sign(x, z)$
3. $checksign(sign(x, y), pk(y)) = x$

Our algorithm also terminates on this equational theory, as shown in Appendix B.

5.2.5 Addition

The following convergent equational theory $\mathcal{E}_{add}$ is a simple model of addition introduced and was proved decidable in [1]:

1. $plus(x, s(y)) = plus(s(x), y)$
2. $plus(x, 0) = x$
3. $pred(s(x)) = x$

In Appendix B we show that our algorithm terminates on this equational theory as well.

5.3 Going beyond with fair strategies

In [1] decidability is also shown for an equational theory modeling homomorphic encryption. For our procedure to terminate on this theory we use a particular saturation strategy.

Homomorphic encryption.

The theory $\mathcal{E}_{\mathsf{hom}}$ of homomorphic encryption that has been studied in [1,10] is as follows:

$$fst(pair(x, y)) = x \qquad snd(pair(x, y)) = y \qquad dec(enc(x, y), y) = x$$
$$enc(pair(x, y), z) = pair(enc(x, z), enc(y, z))$$
$$dec(pair(x, y), z) = pair(dec(x, z), dec(y, z))$$

In general, our algorithm does not terminate under this equational theory. Consider for instance the frame $\phi = \nu a, b.\{w_1 \mapsto pair(a, b)\}$. We have that:

$$\mathrm{Init}(\varphi) \supseteq \{[w_1 \rhd pair(a, b)],\ [enc(X, Y) \rhd enc(x, y) \ \mid\ X \rhd x,\ Y \rhd y]\}.$$

As in Example 11 we can obtain an unbounded number of solved facts whose projections are of the form:

$$[pair(enc(\dots enc(a, z_1)\dots, z_n), enc(\dots enc(b, z_1)\dots, z_n)) \mid z_1, \dots, z_n].$$

However, we can guarantee termination by using a *fair* saturation strategy. We say that a saturation strategy is fair if whenever a rule instance is enabled it will eventually be taken. Indeed in the above example using a fair strategy we will eventually add the facts $[fst(w_1) \rhd a]$ and $[snd(w_1) \rhd b]$. Now the "problematic" facts described above become redundant and are not added to the knowledge base anymore. One may note that a fair strategy does not guarantee termination in Example 11 (intuitively, because the function g is one-way and a is not deducible in that example).

The proof of termination will as for the previous theories define functions $\mathcal{Q}$, $\mathsf{m_f}$ and $\mathsf{m_e}$. The main argument of the proof is the observation that due to fairness only a finite number of solved facts not in $\mathcal{Q}(\varphi)$ can be added. More details are given in Appendix B.

6 Implementation

With certain optimizations described below, our saturation algorithm runs in polynomial time for subterm convergent equational theories, $\mathcal{E}_{mal}$, $\mathcal{E}_{blind}$, and $\mathcal{E}_{td}$.

6.1 Optimizations

Deciding generation in polynomial time ($\mathsf{F} \vdash t$*).*The recursive algorithm obtained immediately from the generation rules is not polynomial. However, by using memoization, its complexity becomes polynomial. Using the same trick, we can compute a recipe R such that $\mathsf{F} \vdash^R t$ in polynomial time, if we store R in DAG form.

*Recipes in DAG form.*Indeed, as shown by the following example, any recipe might grow to an exponential size if it is not stored in DAG form.

Example 12 (from [10]) Consider the theory $\mathcal{E}_{\mathsf{DY}}$ described below:

$$\mathcal{E}_{\mathsf{DY}} = \{dec(enc(x, y), y) = x,\ proj_1(\langle x, y\rangle) = x,\ proj_2(\langle x, y\rangle) = y\}$$

and the two families of frames:

- $\varphi_n = \{w_1 \mapsto t_n^0,\ w_2 \mapsto c_0,\ w_3 \mapsto c_1\}$, and
- $\varphi_n' = \{w_1 \mapsto t_n^1,\ w_2 \mapsto c_0,\ w_3 \mapsto c_1\}$,

where $t_0^i = c_i$ and $t_{n+1}^i = \langle enc(t_n^i, k_n^i), k_n^i\rangle$, $i \in \{0, 1\}$. This example shows that the non-DAG size of the recipes needed to distinguish the frames increases exponentially, while the DAG size grows only linearly. Indeed, the test required to distinguish between φ_n and φ_n' is $R_n \overset{?}{\sim} w_2$, where $R_0 = w_1$ and $R_{n+1} = dec(proj_1(R_n), proj_2(R_n))$.

Therefore, we require that the term R in $[R \rhd u \mid \Delta]$ and the terms U and V in $[U \sim V \mid \Delta]$ are stored in DAG form.

*Optimization to solve ground side conditions.*Using different combinations of solved facts to solve ground side conditions is unnecessary work. Therefore we consider that the standard F-Solving and E-Solving rules are applied only when the side condition being solved contains at least one variable. To solve a side condition of the form $X \rhd t$ when t is ground, we use the two rules described in Figure 2. Again, as for $\oplus$, we suppose that the choice of recipes N and M is uniform.

F-Solving'

$$\frac{\begin{array}{l}\mathsf{f}_1 = [M \rhd t \mid X \rhd u, \dots, X_k \rhd t_k],\ \mathrm{var}(t_0) = \emptyset \\ \mathsf{F} \vdash^N u,\ \mathrm{var}(N) \cap \mathrm{var}(\mathsf{f}_1) = \emptyset\end{array}}{(\mathsf{F}, \mathsf{E}) \Longrightarrow (\mathsf{F}, \mathsf{E}) \oplus \mathsf{f}_0}$$

where $\mathsf{f}_0 = [M\{X \mapsto N\} \rhd t \mid X_1 \rhd t_1, \dots, X_k \rhd t_k]$.

E-Solving'

$$\frac{\begin{array}{l}\mathsf{f}_1 = [U \sim V \mid Y \rhd s, X_1 \rhd t_1, \dots, X_k \rhd t_k] \in \mathsf{E},\ \mathrm{var}(s) = \emptyset \\ \mathsf{F} \vdash^M s,\ \mathrm{var}(M) \cap \mathrm{var}(\mathsf{f}_1) = \emptyset\end{array}}{(\mathsf{F}, \mathsf{E}) \Longrightarrow (\mathsf{F}, \mathsf{E} \cup \{\mathsf{f}_0\})}$$

where $\mathsf{f}_0 = [U\{Y \mapsto M\} \sim V\{Y \mapsto M\} \mid \{X_i \rhd t_i\}_{1 \le i \le k}]$.

Fig. 2 Optimized saturation rules for solving ground side conditions

The soundness of this optimization is assured by Lemma 5 (whose proof is immediate) whereas completeness is shown by proving Lemma 3 and Lemma 4 in the context of the new saturation rules.

Lemma 5 (soundness of the two additional rules) *Let φ be a frame and (F, E) be a knowledge base such that every fact in (F, E) holds in φ. Let f_1 and f_0 be two facts as in rules F-Solving' (resp. E-Solving'). If f_1 holds in φ then f_0 holds in φ.*

Lemma 3 *Let (F, E) be a saturated knowledge base and $\mathsf{f} = [U \sim V \mid X_1 \rhd t_1, \dots, X_k \rhd t_k]$ be an equational fact in E. For any substitution σ grounding for $\{t_1, \dots, t_k\}$ such that $\mathsf{F} \vdash t_i\sigma$ $(1 \le i \le k)$, we have that $\mathsf{F} \vdash^{R_i} t_i\sigma$ for some R_i $(1 \le i \le k)$ and $\mathsf{E} \models U\tau \sim V\tau$ where $\tau = \{X_1 \mapsto R_1, \dots, X_k \mapsto R_k\}$.*

Proof By induction on $\sum_{i=1}^{k} |t_i\sigma|$. We distinguish two cases:

1. f *is a solved equational fact.* The proof is as before.
2. f *is an unsolved equational fact.* In such a case, there exists t_j such that $t_j \notin \mathcal{X}$. Let us assume w.l.o.g. that $j = 1$. If t_1 is not ground, then the proof is as before. If t_1 is ground and because (F, E) is saturated,

$$\mathsf{f}_2 = [U\{X_1 \mapsto M\} \sim V\{X_1 \mapsto M\} \mid X_2 \rhd t_2, \dots, X_k \rhd t_k]$$

must be in E by rule E-Solving', where M is such that $\mathsf{F} \vdash^M t_1$.
We can apply the induction hypothesis on the fact f_2 and the same substitution σ to obtain that there exist R_i $(i \ge 2)$ such that $\mathsf{F} \vdash^{R_i} t_i\sigma$ and:

$$\mathsf{E} \models (U \sim V)\{X_1 \mapsto M\}\{X_2 \mapsto R_2, \dots, X_k \mapsto R_k\}$$

We chose R_1 and M and we immediately obtain the conclusion. □

Lemma 4 *Let* (F, E) *be a saturated knowledge base. Let* $\mathsf{f} = [R \rhd t \mid X_1 \rhd t_1, \ldots, X_k \rhd t_k]$ *be a deduction fact such that* $(\mathsf{F}, \mathsf{E}) \oplus \mathsf{f} = (\mathsf{F}, \mathsf{E})$. *For any substitution* σ *grounding for* $\{t_1, \ldots, t_k\}$ *such that* $\mathsf{F} \vdash t_i\sigma$ $(1 \leq i \leq k)$, *we have that there exist* $R_1, \ldots, R_k$ *and* W *such that*

- $\mathsf{F} \vdash^W t\sigma$, *and* $\mathsf{F} \vdash^{R_i} t_i\sigma$ *for* $1 \leq i \leq k$;
- $\mathsf{E} \models W \sim R\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}$.

Proof By induction on $\sum_{i=1}^{k} |t_i\sigma|$. We distinguish two cases. If f is solved, the proof is as before. If f is not solved, there exists j such that $t_j \not\in \mathcal{X}$. We assume w.l.o.g. that $j = 1$. If t_1 contains at least one variable, the proof is as before. Otherwise, if t_1 is ground and because (F, E) is saturated, rule F-Solving' must have been applied and therefore we can apply the induction hypothesis on

$$\mathsf{f}_2 = [R\{X_1 \mapsto N\} \rhd t \mid X_2 \rhd t_2, \ldots, X_k \rhd t_k]$$

(where N is such that $F \vdash^N t_1$) and on the same substitution σ to obtain that there exist R_i $(i \geq 2)$ and W such that

- $\mathsf{F} \vdash^W t\sigma$ and $F \vdash^{R_i} t_i\sigma$, for $2 \leq i \leq k$
- $\mathsf{E} \models R\{X_1 \mapsto N\}\{X_2 \mapsto R_2, \ldots, X_k \mapsto R_k\} \sim W$

We choose $R_1 = N$ and we immediately obtain our conclusion. □

6.2 Complexity

Theorem 2 *Using the optimizations described in Section 6.1, and if* φ *is in normal form, the saturation algorithm terminates in polynomial time for any subterm convergent equational theory, for* $\mathcal{E}_{td}$, *for* $\mathcal{E}_{mal}$ *and for* $\mathcal{E}_{blind}$.

In the remaining, we consider an equational theory $\mathcal{E}$ that is either subterm convergent, or $\mathcal{E} \in \{\mathcal{E}_{mal}, \mathcal{E}_{blind}, \mathcal{E}_{td}\}$. We define the following set:

$$\mathcal{Q}(\varphi) = \{[r\sigma \mid t_1, \ldots, t_k]\}$$

for every rewrite rule $l \to r$, for every partial substitution $\sigma : \mathrm{var}(l) \to \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ and for every set of incomparable positions $p_1, \ldots, p_k \in \mathrm{pos}(l)$ such that for every i $(1 \leq i \leq k)$ we have that $t_i = (l|_{p_i})\sigma$.

In order to prove Theorem 2, we need an additional lemma.

Lemma 6 *Let* φ *be a frame and* (F, E) *be such that* $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$. *For any unsolved deduction fact* $\mathsf{f} \in \mathsf{F}$ *we have that* $\hat{\mathsf{f}} \in \mathcal{Q}(\varphi)$.

Proof First, note that an unsolved deduction fact obtained by applying Narrowing on a solved fact satisfies this property. Now assume we have an unsolved deduction fact $\hat{\mathsf{f}} = [r\sigma \mid (l|_{p_1})\sigma, \ldots, (l|_{p_k})\sigma] \in \mathcal{Q}(\varphi)$ and assume one of its side conditions $(l|_{p_i})\sigma$ is being solved. Assume w.l.o.g. that $i = 1$.

- If $(l|_{p_1})\sigma$ is ground, rule F-Solving' must be applied. We therefore obtain a fact $\hat{\mathsf{f}}' = [r\sigma \mid (l|_{p_2})\sigma, \ldots, (l|_{p_k})\sigma]$.

- If $(l|_{p_1})\sigma$ is not ground, rule F-Solving is applied and $l|_{p_1}$ is necessarily not a variable (by the definition of σ, it maps variables only to ground terms). Therefore $l|_{p_1}$ is of the form $g(s_1, \dots, s_l)$ for some function symbol $g \in \mathcal{F}$. We distinguish three cases:
  - If the side condition is solved using a deduction fact whose projection is of the form $[t \mid \emptyset]$ for some $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$, let $\sigma' = \mathrm{mgu}((l|_{p_1})\sigma, t)$ and consider $\tau = \sigma \circ \sigma'$. By rule F-Solving, the side condition $(l|_{p_1})\sigma$ will be replaced by side conditions $((l|_{p_1})|_{q_j})\tau$, for all $(l|_{p_1})|_{q_j} \in \mathcal{X}$ and therefore the fact resulting from the application of the rule satisfies the property.
  - If the side condition is solved using a fact whose projection is of the form $[g(x_1, \dots, x_l) \mid x_1, \dots, x_l]$, then the side condition $(l|_{p_1})\sigma$ will be replaced by side conditions $(l|_{p_1 \cdot j})\sigma$, for $1 \le j \le l$.
  - If the side conditions is solved using a "special" fact $[sign(t, x) \mid x]$ (with $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$), $[enc(x, t) \mid x]$ (with $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$), $[td(t_1, t_2, t_3)]$ (with $t_1, t_2, t_3 \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$) or $[f(t_1, t_2, t_3, x) \mid x]$ (with $t_1, t_2, t_3 \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$), we obtain by a case-by-case analysis that the property is satisfied by the resulting fact. $\square$

Now, we are able to prove Theorem 2

Proof (of Theorem 2)

We first show that any knowledge base contains a polynomial number of deduction facts. Indeed, there are a polynomial number of solved deduction facts. Applying rule Narrowing yields a polynomial number of unsolved deduction facts. We also know, thanks to Lemma 6, that for any frame φ (in normal form), for any (F, E) reachable from $\mathrm{Init}(\varphi)$, and for any unsolved fact $\mathsf{f} \in \mathsf{F}$, we have that $\hat{\mathsf{f}} \in \mathcal{Q}(\varphi)$.

We consider the two following orders:

- the order $<_p$ defined on sets of positions as follows:

$$\{p_0, \dots, p_\ell\} <_p \{q_1, \dots, q_k, p_1, \dots, p_\ell\} \text{ iff } q_1, \dots, q_k \text{ are incomparable positions and } p_0 \text{ is a prefix of } q_i \ (1 \le i \le k).$$

- the order $<_\mathsf{f}$ defined on deduction facts whose projection are in $\mathcal{Q}(\varphi)$:

$$\mathsf{f}_0 <_\mathsf{f} \mathsf{f}_1 \text{ iff either } \ell < k \text{ or } \ell = k \text{ and } \{p_1, \dots, p_k\} <_p \{p'_1, \dots, p'_\ell\}.$$

where $\mathsf{f}_0 = [R \rhd r\sigma \mid X_1 \rhd l|_{p_1}\sigma, \dots, X_k \rhd l|_{p_k}]$, and
$\mathsf{f}_1 = [R' \rhd r\sigma' \mid X_1 \rhd l|_{p'_1}\sigma', \dots, X_l \rhd l|_{p'_\ell}\sigma']$.

As $<_\mathsf{f}$ does not depend on the frame, all strictly decreasing sequences of deduction facts have at most a constant size. Also note that if f_1 and f_0 are as in rule F-Solving or F-Solving', we have that $\mathsf{f}_0 <_\mathsf{f} \mathsf{f}_1$. There are at most a polynomial number of choices to be made when solving each deduction fact (which side condition, which solved deduction fact). As the resulting facts will be smaller (according to $<_\mathsf{f}$) than the initial fact, and as any such sequence has at most a constant length, an unsolved fact will generated at most a polynomial number of facts.

We now show that each deduction fact has at most a polynomial size if the recipes are stored in DAG form. This is obviously true of the initial facts. The other recipes are obtained from the initial recipes by applying a polynomial number of substitutions whose size is polynomially bounded. Therefore all recipes have polynomial size.

It remains to show that there are a polynomial number of equational facts. This is true of the (necessarily solved) equational facts added during application of Narrowing and F-Solving (via the $\oplus$ operation). The other possibility to generate equational facts is Unifying, which generates a polynomial number of (possible unsolved) equational facts. All such unsolved equational facts have side conditions which are either ground or variables. Therefore, each such unsolved equational fact will lead to at most a polynomial number of other equational facts by applying rule E-Solving'. □

6.3 The KiSs tool

A C++ implementation of the procedures described in this paper is provided in the KiSs (Knowledge In Security protocolS) tool [16].

The tool implements a partially fair saturation strategy and a uniform $\oplus$. The fairness employed by the tool is sufficient to decide the theory $\mathcal{E}_{\mathsf{hom}}$. Moreover the tool implements the optimizations described in subsection 6.1. This makes the procedure terminate in polynomial time for subterm convergent equational theories, and the theories $\mathcal{E}_{blind}$, $\mathcal{E}_{mal}$ and $\mathcal{E}_{td}$.

The performances of the tool are comparable to the YAPA tool [9,10] and on most examples the tool terminates in less than a second. In [10] a family of contrived examples is presented to diminish the performance of YAPA, exploiting the fact that YAPA does not implement DAG representations of terms and recipes, as opposed to KiSs. As expected, KiSs indeed performs better on these examples.

In [10] a class of equational theories for which YAPA terminates is identified and it is not known whether our procedure terminates on this specific class. However, we have shown that our procedure terminates on all examples of equational theories presented in [10]. This requires to prove termination of our saturation procedure for each equational theory presented in [10]. In addition, our tool terminates on the theories $\mathcal{E}_{mal}$ and $\mathcal{E}_{td}$ whereas YAPA does not. Of course, YAPA may also terminate on examples outside the class exhibited in [10]. Hence the question whether termination of our procedures encompasses termination of YAPA is still open.

7 Conclusion and future work

We have proposed and implemented a procedure for deduction and for static equivalence for convergent equational theories. Our procedure terminates for a wide range of equational theories. In particular, we obtain a new decidability result for the theory of trapdoor commitment.

All of our examples feature convergent term rewriting systems which are right-linear. Even though it is unlikely that a non-right-linear term rewriting system is useful for modeling cryptographic primitives, we note that this is not an inherent limitation of our procedure, as illustrated by the following (contrived) rewrite rule

$$g(x) \to f(x,x)$$

for which our procedure terminates.

Our procedure however does not terminate in general on the following equational theories modelling re-encryption:

$$renc(enc(x,y,z),t) \to enc(x,y,f(z,t))$$

as illustrated below. Starting from the frame

$$\varphi = \nu a, b, c.\{w_1 \mapsto enc(a, b, c)\}$$

our knowledge base will contain the following infinite set of deduction facts:

$$\begin{array}{lcccl}
[& w_1 & \rhd & enc(a, b, c) & \mid \emptyset] \\
[& renc(w_1, X_1) & \rhd & enc(a, b, f(c, x_1)) & \mid X_1 \rhd x_1] \\
[& renc(renc(w_1, X_1), X_2) & \rhd & enc(a, b, f(f(c, x_1), x_2) & \mid X_1 \rhd x_1, X_2 \rhd x_2] \\
 & & \ldots & &
\end{array}$$

As future work, we indent to extend our approach in order to handle the case of re-encryption and the case of associative commutative operators (like xor), which cannot be handled by a convergent term rewriting system.

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 ACM Symposium on Principles of Programming Languages (POPL'01)*. ACM, 2001.
3. S. Anantharaman, P. Narendran, and M. Rusinowitch. Intruders with caps. In *Proc. 18th International Conference on Term Rewriting and Applications (RTA'07)*, volume 4533 of *LNCS*. Springer, 2007.
4. A. Armando et al. The AVISPA Tool for the automated validation of internet security protocols and applications. In *Proc. 17th Int. Conference on Computer Aided Verification (CAV'05)*, volume 3576 of *LNCS*, pages 281–285. Springer, 2005.
5. M. Arnaud, V. Cortier, and S. Delaune. Combining algorithms for deciding knowledge in security protocols. In F. Wolter, editor, *Proceedings of the 6th International Symposium on Frontiers of Combining Systems (FroCoS'07)*, volume 4720 of *Lecture Notes in Artificial Intelligence*, pages 103–117, Liverpool, UK, Sept. 2007. Springer.
6. M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proc. 21st IEEE Computer Security Foundations Symposium (CSF'08)*, 2008.
7. M. Backes, M. Maffei, and D. Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *Proc. IEEE Symposium on Security and Privacy (S&P'08)*. IEEE Comp. Soc. Press, 2008.
8. M. Baudet. Deciding security of protocols against off-line guessing attacks. In *12th ACM Conference on Computer and Communications Security (CCS'05)*, 2005.
9. M. Baudet. YAPA (Yet Another Protocol Analyzer), 2008. `http://www.lsv.ens-cachan.fr/~baudet/yapa/index.html`.
10. M. Baudet, V. Cortier, and S. Delaune. YAPA: A generic tool for computing intruder knowledge. In R. Treinen, editor, *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA'09)*, volume 5595 of *Lecture Notes in Computer Science*, pages 148–163, Brasília, Brazil, June-July 2009. Springer.
11. M. Berrima, N. Ben Rajeb, and V. Cortier. Deciding knowledge in security protocols under some e-voting theories. Research Report RR-6903, INRIA, April 2009.
12. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Comp. Soc. Press, 2001.
13. B. Blanchet, M. Abadi, and C. Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *Symposium on Logic in Computer Science*, pages 331–340. IEEE Comp. Soc. Press, 2005.
14. Y. Chevalier. *Résolution de problèmes d' accessibilité pour la compilation et la validation de protocoles cryptographiques.* PhD thesis, Université Henri Poincaré, Nancy (France), 2003.

15. Y. Chevalier and M. Kourjieh. Key substitution in the symbolic analysis of cryptographic protocols. In *Proc. 27th International Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'07)*, pages 121–132, 2007.
16. Ş. Ciobâcă. KiSs, 2009. `http://www.lsv.ens-cachan.fr/~ciobaca/kiss`.
17. Ş. Ciobâcă, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. In R. Schmidt, editor, *Proceedings of the 22nd International Conference on Automated Deduction (CADE'09)*, Lecture Notes in Artificial Intelligence, pages 355–370, Montreal, Canada, Aug. 2009. Springer.
18. R. Corin, J. Doumen, and S. Etalle. Analysing password protocol security against off-line dictionary attacks. In *Proc. 2nd International Workshop on Security Issues with Petri Nets and other Computational Models (WISP'04)*, ENTCS, 2004.
19. V. Cortier and S. Delaune. Deciding knowledge in security protocols for monoidal equational theories. In *Proc. 14th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, LNAI. Springer, 2007.
20. V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.
21. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.
22. S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *14th European Symposium on Programming (ESOP'05)*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.
23. P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for the equational theory of Abelian groups with distributive encryption. *Information and Computation*, 205(4):581–623, 2007.
24. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*, 2001.
25. T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. 5th Int. Security Protocols Workshop*, volume 1361 of *LNCS*. Springer, 1997.
26. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299:451–475, 2003.

A Proofs of Section 4

A.1 Soundness

Lemma 7 *Let φ be a frame and (F, E) be a knowledge base such that every fact in (F, E) (deduction or equational) holds in φ. Let f_0 be a fact that holds in φ, then every fact in $(\mathsf{F}, \mathsf{E}) \oplus \mathsf{f}_0$ holds in φ.*

Lemma 1 *Let φ be a frame and (F, E) be a knowledge base such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$. Then every $\mathsf{f} \in \mathsf{F} \cup \mathsf{E}$ holds in φ.*

Proof By induction on the derivation $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$.

Base case: We have that $(\mathsf{F}, \mathsf{E}) = \mathrm{Init}(\varphi)$. To conclude, we have to show that the facts and the equations we put in the initial knowledge base hold in φ.

There are three kind of deduction facts that can be added in the knowledge base: the facts that come from φ, those of the form $[n \rhd n]$ for $n \in \mathrm{fn}(\varphi)$, and those of the form:

$$[f(X_1, \dots, X_k) \rhd f(x_1, \dots, x_k) \mid X_1 \rhd x_1, \dots, X_k \rhd x_k].$$

It is easy to see that all these facts hold in φ and we can conclude by Lemma 7.

Induction step: In such a case, we have $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}', \mathsf{E}') \Longrightarrow (\mathsf{F}, \mathsf{E})$. We perform a case analysis on the inference rule used in $(\mathsf{F}', \mathsf{E}') \Longrightarrow (\mathsf{F}, \mathsf{E})$. For each rule, we show that the resulting fact f_0 holds in φ and we conclude by relying on Lemma 7.

Rule Narrowing: Let $\mathsf{f} = [M \rhd C[t] \mid X_1 \rhd x_1, \dots, X_k \rhd x_k]$ be the deduction fact, $l \to r \in \mathcal{R}_\mathcal{E}$ be the rewrite rule and $\sigma = \mathrm{mgu}(l, t)$ be the substitution involved in this step. Let $\mathsf{f}_0 = [M \rhd (C[r])\sigma \mid X_1 \rhd x_1\sigma, \dots, X_k \rhd x_k\sigma]$ be the resulting deduction fact.

We show that f_0 holds in φ. Let τ be a substitution such that $\varphi \vdash_\mathcal{E} x_i\sigma\tau$ with recipe M_i $(1 \leq i \leq k)$. Since f holds in φ, we have that $\varphi \vdash_\mathcal{E} (C[t])\sigma\tau$ with recipe $M' = M\{X_1 \mapsto M_1, \dots, X_k \mapsto M_k\}$. It is easy to see that the following equalities are satisfied:

$$(C[t])\sigma\tau = (C[l])\sigma\tau =_\mathcal{E} (C[r])\sigma\tau$$

Therefore $\varphi \vdash_\mathcal{E} (C[r])\sigma\tau$ by recipe M', and thus f_0 holds in φ.

Rule F-Solving: Let $\mathsf{f}_1 = [M \rhd t \mid X_0 \rhd t_0, \dots, X_k \rhd t_k]$ with $t_0 \notin \mathcal{X}$ and $\mathsf{f}_2 = [N \rhd s \mid Y_1 \rhd y_1, \dots, Y_\ell \rhd y_\ell]$ be the two deduction facts and $\sigma = \mathrm{mgu}(s, t_0)$ be the substitution involved in this step. Let f_0 be the resulting deduction fact:

$$\mathsf{f}_0 = [M\{X_0 \mapsto N\} \rhd t\sigma \mid X_1 \rhd t_1\sigma, \dots, X_k \rhd t_k\sigma, Y_1 \rhd y_1\sigma, \dots, Y_\ell \rhd y_\ell\sigma].$$

We show that f_0 holds in φ. Let τ be a substitution such that $\varphi \vdash_\mathcal{E} t_i\sigma\tau$ with recipe M_i $(1 \leq i \leq k)$ and $\varphi \vdash_\mathcal{E} y_j\sigma\tau$ with recipes N_j $(1 \leq j \leq \ell)$. Since f_2 holds in φ, we have that $\varphi \vdash_\mathcal{E} s\sigma\tau$ with recipe $N' = N\{Y_1 \mapsto N_1, \dots, Y_\ell \mapsto N_\ell\}$. Since f_1 holds in φ and $s\sigma\tau = t_0\sigma\tau$, we deduce that $\varphi \vdash_\mathcal{E} t\sigma\tau$ with recipe

$$\begin{aligned} &M\{X_0 \mapsto N', X_1 \mapsto M_1, \dots, X_k \mapsto M_k\} \\ &= (M\{X_0 \mapsto N\})\{X_1 \mapsto M_1, \dots, X_k \mapsto M_k, Y_1 \mapsto N_1, \dots, Y_\ell \mapsto N_\ell\}. \end{aligned}$$

This allows us to conclude that f_0 holds in φ.

Rule Unifying: Let $\mathsf{f}_1 = [M \rhd t \mid X_1 \rhd x_1, \dots, X_k \rhd x_k]$ and $\mathsf{f}_2 = [N \rhd s \mid Y_1 \rhd y_1, \dots, Y_\ell \rhd y_\ell]$ be the two solved deduction facts and $\sigma = \mathrm{mgu}(s, t)$ be the substitution involved in this step. Let f_0 be the resulting equational fact:

$$\mathsf{f}_0 = [M \sim N \mid X_1 \rhd x_1\sigma, \dots, X_k \rhd x_k\sigma, Y_1 \rhd y_1\sigma, \dots, Y_\ell \rhd y_\ell\sigma].$$

We show that f_0 holds in φ. Let τ be a substitution such that $\varphi \vdash_\mathcal{E} x_i\sigma\tau$ with recipe M_i $(1 \leq i \leq k)$ and $\varphi \vdash_\mathcal{E} y_j\sigma\tau$ with recipes N_j $(1 \leq j \leq \ell)$. Since f_1 and f_2 holds in φ and

$s\sigma\tau = t\sigma\tau$, we deduce that $\varphi \vdash_{\mathcal{E}} t\sigma\tau$ with recipe $M\{X_1 \mapsto M_1, \dots, X_k \mapsto M_k\}$ and $N\{Y_1 \mapsto N_1, \dots, Y_k \mapsto N_\ell\}$. This allows us to conclude that f_0 holds in φ.

Rule E-Solving: Let $\mathsf{f}_1 = [U \sim V \mid Y \triangleright s, X_1 \triangleright t_1, \dots, X_k \triangleright t_k]$ be the equational fact and $\mathsf{f}_2 = [N \triangleright t \mid Y_1 \triangleright y_1, \dots, Y_\ell \triangleright y_\ell]$ be the solved deduction fact, and $\sigma = \mathrm{mgu}(s,t)$ be the substitution involved in this step. Let f_0 be the resulting equational fact:

$$\mathsf{f}_0 = [U\{Y \mapsto N\} \sim V\{Y \mapsto N\} \mid X_1 \triangleright t_1\sigma, \dots, X_k \triangleright t_k\sigma, Y_1 \triangleright y_1\sigma, \dots, Y_\ell \triangleright y_\ell\sigma].$$

We show that f_0 holds in φ. Let τ be a substitution such that $\varphi \vdash_{\mathcal{E}} t_i\sigma\tau$ with recipe M_i $(1 \leq i \leq k)$ and $\varphi \vdash_{\mathcal{E}} y_j\sigma\tau$ with recipe N_j $(1 \leq j \leq \ell)$. Since f_2 holds in φ, we deduce that $\varphi \vdash_{\mathcal{E}} t\sigma\tau$ with recipe $N' = N[Y_1 \mapsto N_1, \dots, Y_\ell \mapsto N_\ell]$. Since $s\sigma\tau = t\sigma\tau$, we deduce that $\varphi \vdash_{\mathcal{E}} s\sigma\tau$ with recipe N', and by using the fact that f_1 holds in φ we deduce that

$$(U\{Y \mapsto N', X_1 \mapsto M_1, \dots, X_k \mapsto M_k\} =_{\mathcal{E}} V\{Y \mapsto N', X_1 \mapsto M_1, \dots, X_k \mapsto M_k\})\varphi.$$

Thus, f_0 holds in φ. □

A.2 Completeness

Lemma 3 *Let* (F, E) *be a saturated knowledge base and* $\mathsf{f} = [U \sim V \mid X_1 \triangleright t_1, \dots, X_k \triangleright t_k]$ *be an equational fact in* E. *For any substitution* σ *grounding for* $\{t_1, \dots, t_k\}$ *such that* $\mathsf{F} \vdash t_i\sigma$ $(1 \leq i \leq k)$, *we have that* $\mathsf{F} \vdash^{R_i} t_i\sigma$ *for some* R_i $(1 \leq i \leq k)$ *and* $\mathsf{E} \models U\tau \sim V\tau$ *where* $\tau = \{X_1 \mapsto R_1, \dots, X_k \mapsto R_k\}$.

Proof We show this result by induction on $\sum_{i=1}^{k} |t_i\sigma|$. We distinguish two cases:

1. f *is a solved equational fact*, i.e. $t_1, \dots, t_k$ are variables (not necessarily distinct), say $x_1, \dots, x_k$. In such a case, we have that
$$\mathsf{E} \models U\{X_1 \mapsto x_1, \dots, X_k \mapsto x_k\} \sim V\{X_1 \mapsto x_1, \dots, X_k \mapsto x_k\}.$$
We choose each R_i arbitrarily such that $x_i = x_j$ implies $R_i = R_j$. Then, it is easy to conclude.
2. f *is an unsolved equational fact.* In such a case, there exists t_j such that $t_j \notin \mathcal{X}$. Let us assume w.l.o.g. that $j = 1$. As $\mathsf{F} \vdash t_1\sigma$, we know that there exist a solved deduction fact $\mathsf{f}^1 = [R^1 \triangleright t^1 \mid X_1^1 \triangleright x_1^1, \dots, X_\ell^1 \triangleright x_\ell^1]$ in F and a substitution τ such that $t^1\tau = t_1\sigma$ and $\mathsf{F} \vdash^{R_i'} x_i^1\tau$ $(1 \leq i \leq \ell)$.
Let $\rho = \mathrm{mgu}(t_1, t^1)$. We have that the following fact f_2 is in E since (F, E) is saturated:
$$[U\{X_1 \mapsto R^1\} \sim V\{X_1 \mapsto R^1\} \mid X_1^1 \triangleright x_1^1\rho, \dots, X_\ell^1 \triangleright x_\ell^1\rho, X_2 \triangleright t_2\rho, \dots, X_k \triangleright t_k\rho].$$
Let σ' be the substitution such that $\sigma \cup \tau = \rho \circ \sigma'$. As the fact f^1 is solved, $x_1^1\rho\sigma', \dots, x_\ell^1\rho\sigma'$ are strict subterms of $t^1\rho\sigma' = t^1\tau$ and $\sum_{i=1}^{\ell} |x_i^1\rho\sigma'| < |t^1\tau| = |t_1\sigma|$. Thus we can apply our induction hypothesis on the equational fact f_2 with the substitution σ'. This allows us to obtain that there exist $M_1^1, \dots, M_\ell^1, M_2, \dots, M_k$ such that $\mathsf{F} \vdash^{M_i} t_i\rho\sigma' = t_i\sigma$ $(2 \leq i \leq k)$ and $\mathsf{F} \vdash^{M_i^1} x_i^1\rho\sigma' = x^1\sigma$ $(1 \leq i \leq \ell)$ and the following equation $(\star)$
$$\begin{aligned}\mathsf{E} \models\ & (U\{X_1 \mapsto R^1\})\{X_1^1 \mapsto M_1^1, \dots, X_\ell^1 \mapsto M_\ell^1, X_2 \mapsto M_2, \dots, X_k \mapsto M_k\} \\ & \sim \\ & (V\{X_1 \mapsto R^1\})\{X_1^1 \mapsto M_1^1, \dots, X_\ell^1 \mapsto M_\ell^1, X_2 \mapsto M_2, \dots, X_k \mapsto M_k\}\end{aligned}$$
We choose $R_1 = R^1\{X_1^1 \mapsto M_1^1, \dots, X_\ell^1 \mapsto M_\ell^1\}$ and $R_2 = M_2, \dots, R_k = M_k$. Thus, the equation $(\star)$ can be rewritten as follows:
$$\mathsf{E} \models U\{X_1 \mapsto R_1, \dots, X_k \mapsto R_k\} \sim V\{X_1 \mapsto R_1, \dots, X_k \mapsto R_k\}.$$
This allows us to conclude. □

Lemma 8 *Let* (F,E) *be a knowledge base and* t *be a term in* $\mathcal{T}(\mathcal{F},\mathcal{N}\cup\mathcal{X})$. *Let* σ *be a grounding substitution for* t. *If* $\mathsf{F} \vdash^{W} t$ *and* $\mathsf{F} \vdash^{R_x} x\sigma$ *for every* $x \in \mathrm{var}(t)$, *then* $\mathsf{F} \vdash^{W'} t\sigma$ *where* $W' = W\{x \mapsto R_x\}_{x\in\mathrm{var}(t)}$.

Proof We show this result by induction on $|t|$.
Base case: $|t| = 0$, *i.e.* t *is a variable, say* x. As $\mathsf{F} \vdash^{W} t$, it follows that $W = t = x$. By hypothesis, there exists R such that $\mathsf{F} \vdash^{R} x\sigma = t\sigma$. This allows us to conclude.

Induction case: $|t| > 0$. As $\mathsf{F} \vdash^{W} t$, it follows that there exist a fact $\mathsf{f} \in \mathsf{F}$ and a substitution τ such that:

- $\mathsf{f} = [R \rhd u \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k]$;
- $t = u\tau$;
- $\mathsf{F} \vdash^{R_i} x_i\tau$ for every $1 \leq i \leq k$ and $W = R\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}$.

We have that $\mathrm{var}(u) = \{x_1, \ldots, x_k\}$ and thus, $x_i\tau$ is a strict subterm of $u\tau$ $(1 \leq i \leq k)$. Therefore, we can apply our induction hypothesis on each term $x_i\tau$ with the substitution σ. For each i such that $1 \leq i \leq k$, we obtain that:

$$\mathsf{F} \vdash^{W_i} x_i\tau\sigma \text{ where } W_i = R_i\{x \mapsto R_x\}_{x\in\mathrm{var}(x_i\tau)}.$$

Note that since $t = u\tau$ and $\mathrm{var}(u) = \{x_1, \ldots, x_k\}$, we have that $\mathrm{var}(t) = \mathrm{var}(\{x_1\tau, \ldots, x_k\tau\})$. By using the fact f, we get that $\mathsf{F} \vdash^{W''} u\tau\sigma$ where

$$\begin{aligned} W'' &= R\{X_1 \mapsto R_1\{x \mapsto R_x\}_{x\in\mathrm{var}(t)}, \ldots, X_k \mapsto R_k\{x \mapsto R_x\}_{x\in\mathrm{var}(t)}\} \\ &= (R\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\})\{x \mapsto R_x\}_{x\in\mathrm{var}(t)} \\ &= W\{x \mapsto R_x\}_{x\in\mathrm{var}(t)} \end{aligned}$$

Let $W' = W\{x \mapsto R_x\}_{x\in\mathrm{var}(t)}$, we have that $\mathsf{F} \vdash^{W'} u\tau\sigma$ and since $u\tau\sigma = t\sigma$ we easily conclude. □

Lemma 9 *Let* $\mathsf{f} = [R \rhd t \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k]$ *be a solved fact and* (F,E) *be a knowledge base such that* $(\mathsf{F},\mathsf{E}) \oplus \mathsf{f} = (\mathsf{F},\mathsf{E})$. *Let* σ *be a substitution grounding for* $\{x_1, \ldots, x_k\}$ *such that* $\mathsf{F} \vdash x_i\sigma$ $(1 \leq i \leq k)$. *Then there exist* W *and* R_i $(1 \leq i \leq k)$ *such that:*

- $\mathsf{F} \vdash^{W} t\sigma$, *and* $\mathsf{F} \vdash^{R_i} x_i\sigma$ *for every* $1 \leq i \leq k$;
- $\mathsf{E} \models W \sim R\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}$.

Proof Let f' be the canonical form of f. We first show that $\mathsf{F} \cup \{\mathsf{f}'\} = \mathsf{F}$ implies $\mathsf{F} \vdash t$. This is easily shown by induction on the number of steps to compute the canonical form.

Base case: If f is already in canonical form we have that $\mathsf{f} = \mathsf{f}'$ and hence $\mathsf{F} \vdash t$.
Inductive case: The two rules are of the form

$$\frac{[R \rhd t \mid X_1 \rhd x_1, \ldots, X_k \rhd x_k]}{\mathsf{f}_0 = [R' \rhd t \mid X_1 \rhd x_1, \ldots, X_{i-1} \rhd x_{i-1}, X_{i+1} \rhd x_{i+1}, \ldots, X_k \rhd x_k]}$$

Let f_0' be the canonical form of f_0. By induction hypothesis we have $\mathsf{F} \cup \{\mathsf{f}_0'\} = \mathsf{F}$ implies $\mathsf{F} \vdash t$. As $\mathsf{f}' = \mathsf{f}_0'$ we conclude.

To prove the lemma we consider both cases where f is either useful or redundant.

Useful fact: If f is useful we have that $\mathsf{F} \vdash t$. By what we have just shown, $\mathsf{F} \cup \{\mathsf{f}'\} \neq \mathsf{F}$ which contradicts that $(\mathsf{F},\mathsf{E}) \oplus \mathsf{f} = (\mathsf{F},\mathsf{E})$. Hence, this case is impossible.

Redundant fact: Since $(\mathsf{F},\mathsf{E}) \oplus \mathsf{f} = (\mathsf{F},\mathsf{E})$, it follows that there exists W' such that $\mathsf{F} \vdash^{W'} t$ and $\mathsf{E} \models W' \sim R\{X_1 \mapsto x_1, \ldots, X_k \mapsto x_k\}$. We choose R_i arbitrarily such that $\mathsf{F} \vdash^{R_i} x_i\sigma$. Let $W'' = W'\{x_1 \mapsto R_1, \ldots, x_k \mapsto R_k\}$. Thanks to Lemma 8, we deduce that $\mathsf{F} \vdash^{W''} t\sigma$ and we also have that

$$\mathsf{E} \models (W' \sim R\{X_1 \mapsto x_1, \ldots, X_k \mapsto x_k\})\{x_1 \mapsto R_1, \ldots, x_k \mapsto R_k\},$$

i.e. $\mathsf{E} \models W'' \sim R\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}$.
Let $W = W''$. We have that $\mathsf{F} \vdash^{W} t\sigma$, and $\mathsf{F} \vdash^{R_i} x_i\sigma$ for every $1 \leq i \leq k$. Lastly, we have that $\mathsf{E} \models W \sim R\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}$. □

Lemma 4 *Let* (F,E) *be a saturated knowledge base. Let* $\mathsf{f} = [R \rhd t \mid X_1 \rhd t_1, \ldots, X_k \rhd t_k]$ *be a deduction fact such that* $(\mathsf{F},\mathsf{E}) \oplus \mathsf{f} = (\mathsf{F},\mathsf{E})$*. For any substitution* σ *grounding for* $\{t_1,\ldots,t_k\}$ *such that* $\mathsf{F} \vdash t_i\sigma$ *(*$1 \leq i \leq k$*), we have that there exist* $R_1,\ldots,R_k$ *and* W *such that*

- $\mathsf{F} \vdash^W t\sigma$*, and* $\mathsf{F} \vdash^{R_i} t_i\sigma$ *for* $1 \leq i \leq k$*;*
- $\mathsf{E} \models W \sim R\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}$*.*

Proof We show the result by induction on $\sum_{i=1}^{k} |t_i\sigma|$. We distinguish two cases. If f is solved then we easily conclude by applying Lemma 9.

If f is not solved, there exists j such that $t_j \notin \mathcal{X}$. We assume w.l.o.g. that $j = 1$. Since $\mathsf{F} \vdash t_1\sigma$, there exist a solved deduction fact $\mathsf{f}' \in \mathsf{F}$, some terms $R'_i (1 \leq i \leq \ell)$ and a substitution τ such that:

- $\mathsf{f}' = [R' \rhd t' \mid Y_1 \rhd y_1, \ldots, Y_\ell \rhd y_\ell]$;
- $t'\tau = t_1\sigma$;
- $\mathsf{F} \vdash^{R'_i} y_i\tau$ for every $1 \leq i \leq \ell$.

By application of the $\mathsf{F\text{-}Solving}$ rule to the deduction facts f and f', we obtain the following fact f_0:

$$\mathsf{f}_0 = [R\{X_1 \mapsto R'\} \rhd t\rho \mid X_2 \mapsto t_2\rho, \ldots, X_k \mapsto t_k\rho, Y_1 \mapsto y_1\rho, \ldots, Y_\ell \mapsto y_\ell\rho]$$

where $\rho = mgu(t', t_1)$.

As (F,E) is saturated, $(\mathsf{F},\mathsf{E}) \oplus \mathsf{f}_0 = (\mathsf{F},\mathsf{E})$. Let σ' be the substitution such that $\sigma \cup \tau = \rho \circ \sigma'$. As $y_i\rho\sigma' = y_i(\sigma \cup \tau) = y_i\tau$ are strict disjoint subterms of $t'\tau = t_1\sigma$, it follows that we can apply our induction hypothesis on f_0 and the substitution σ'. Therefore, there exist $R'_2, \ldots, R'_k, R^y_1, \ldots, R^y_\ell$ and W' such that:

- $\mathsf{F} \vdash^{W'} t\rho\sigma'$,
- $\mathsf{F} \vdash^{R'_i} t_i\rho\sigma'$ for every $2 \leq i \leq k$;
- $\mathsf{F} \vdash^{R^y_j} y_j\rho\sigma'$ for every $1 \leq j \leq \ell$;
- $\mathsf{E} \models W' \sim (R\{X_1 \mapsto R'\})\{X_2 \mapsto R'_2, \ldots, X_k \mapsto R'_k, Y_1 \mapsto R^y_1, \ldots, Y_\ell \mapsto R^y_\ell\}$.

Let $W = W'$, $R_1 = R'\{Y_1 \mapsto R^y_1, \ldots, Y_\ell \mapsto R^y_\ell\}$, $R_j = R'_j$ for every $2 \leq j \leq k$. It immediately follows that $\mathsf{E} \models W \sim R\{X_1 \mapsto R_1, \ldots, X_k \mapsto R_k\}$, $\mathsf{F} \vdash^W t\sigma$, and $\mathsf{F} \vdash^{R_i} t_i\sigma$ for $1 \leq i \leq k$. This allows us to conclude. □

A.3 Proof of Theorem 1

Theorem 1 (soundness and completeness) *Let* φ *be a frame and* (F,E) *be a saturated knowledge base such that* $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F},\mathsf{E})$*. Let* $t \in \mathcal{T}(\mathcal{F},\mathcal{N})$ *and* $\mathsf{F}^+ = \mathsf{F} \cup \{[n \rhd n] \mid n \in \mathrm{fn}(t) \smallsetminus \mathrm{bn}(\varphi)\}$*. We have that:*

1. *For all* $M \in \mathcal{T}(\mathcal{F},\mathcal{N} \cup \mathrm{dom}(\varphi))$ *such that* $\mathrm{fn}(M) \cap \mathrm{bn}(\varphi) = \emptyset$*, we have that*
$$M\varphi =_{\mathcal{E}} t \Leftrightarrow \exists N,\ \mathsf{E} \models M \sim N \text{ and } \mathsf{F}^+ \vdash^N t{\downarrow}_{\mathcal{R}_{\mathcal{E}}}$$
2. *For all* $M, N \in \mathcal{T}(\mathcal{F},\mathcal{N} \cup \mathrm{dom}(\varphi))$ *such that* $\mathrm{fn}(M,N) \cap \mathrm{bn}(\varphi) = \emptyset$*, we have*
$$(M =_{\mathcal{E}} N)\varphi \ \Leftrightarrow\ \mathsf{E} \models M \sim N.$$

Proof Let φ be a frame and (F,E) be a saturated knowledge base such that $\text{Init}(\varphi) \Longrightarrow^* (\mathsf{F},\mathsf{E})$.

1.($\Leftarrow$) Let M, N and t be such that $\mathsf{E} \models M \sim N$ and $\mathsf{F}^+ \vdash^N t{\downarrow_{\mathcal{R}_\mathcal{E}}}$. Thanks to Lemma 2, we have that $M\varphi =_\mathcal{E} N\varphi =_\mathcal{E} t$.

($\Rightarrow$) Let M and t be such that $M\varphi =_\mathcal{E} t$.

Let $\mathsf{F}^{++} = \mathsf{F} \cup \{[n \rhd n] \mid n \in \text{fn}(M)\}$. We have that $\mathsf{F}^{++} \vdash^M t_0$ and $t_0 \to^* t{\downarrow_{\mathcal{R}_\mathcal{E}}}$ with $t_0 = M\varphi$.

Let $\{n_1,\ldots,n_\ell\} = \text{fn}(M) \smallsetminus \text{fn}(\varphi \cup \{t\})$. Let $y_1,\ldots,y_\ell$ be fresh variables and $\delta = \{n_1 \mapsto y_1,\ldots,n_\ell \mapsto y_\ell\}$. Let $M' = M\delta$. We have that $\mathsf{F}^{++} \vdash^{M'} t'_0$ and $t'_0 \to^* t{\downarrow_{\mathcal{R}_\mathcal{E}}}$ with $t'_0 = M'\varphi$.

Now, let $\mathsf{E}^{++} = \mathsf{E} \cup \{[n \sim n] \mid n \in \text{fn}(M)\}$. As (F,E) is a saturated knowledge base, we have that $(\mathsf{F}^{++},\mathsf{E}^{++})$ is a saturated knowledge base as well. Now thanks to Proposition 1, we deduce that $\mathsf{E}^{++} \models M \sim M'$, thus $\mathsf{E} \models M \sim M'$ as well.

We show the result by induction on t_0 equipped with the order $<$ induced by the rewriting relation ($t < t'$ if and only if $t' \to^+ t$).

Base case: $\mathsf{F}^+ \vdash^{M'} t_0 = t{\downarrow_{\mathcal{R}_\mathcal{E}}}$. Let $N = M'$, we have $\mathsf{E} \models M \sim N$ and $\mathsf{F} \vdash^N t{\downarrow_{\mathcal{R}_\mathcal{E}}}$.

Induction case: $\mathsf{F}^+ \vdash^{M'} t_0$ *with* $t_0 \neq t{\downarrow_{\mathcal{R}_\mathcal{E}}}$. Let $\mathsf{E}^+ = \mathsf{E} \cup \{[n \sim n] \mid n \in \text{fn}(t) \smallsetminus \text{bn}(\varphi)\}$. We easily see that as (F,E) is a saturated knowledge base we have that $(\mathsf{F}^+,\mathsf{E}^+)$ is a saturated knowledge base as well. Hence we can apply Proposition 2 and deduce that there exist N' and t' such that $\mathsf{F}^+ \vdash^{N'} t'$, $t \to^+_{\mathcal{R}_\mathcal{E}} t'$, and $\mathsf{E}^+ \models M' \sim N'$. It is easy to see that $\mathsf{E} \models M' \sim N'$ as well. We have that $\mathsf{F}^+ \vdash^{N'} t' \to^* t{\downarrow_{\mathcal{R}_\mathcal{E}}}$ and $t' < t_0$. Thus, we can apply our induction hypothesis and we obtain that there exists N such that $\mathsf{E} \models N' \sim N$ and $\mathsf{F}^+ \vdash^N t{\downarrow_{\mathcal{R}_\mathcal{E}}}$.

2.($\Leftarrow$) By Lemma 2, $\mathsf{E} \models M \sim N$ implies $M\varphi =_\mathcal{E} N\varphi$.

($\Rightarrow$) Let M and N such that $M\varphi =_\mathcal{E} N\varphi$. This means that there exists t such that $M\varphi =_\mathcal{E} t$ and $N\varphi =_\mathcal{E} t$. Let $\mathsf{F}^+ = \mathsf{F} \cup \{[n \rhd n] \mid n \in \text{fn}(t) \smallsetminus \text{bn}(\varphi)\}$ and $\mathsf{E}^+ = \mathsf{E} \cup \{[n \sim n] \mid n \in \text{fn}(t) \smallsetminus \text{bn}(\varphi)\}$. By applying 1, we deduce that there exist M', N' such that $\mathsf{E} \models M \sim M'$, $\mathsf{F}^+ \vdash^{M'} t{\downarrow_{\mathcal{R}_\mathcal{E}}}$, $\mathsf{E} \models N \sim N'$ and $\mathsf{F}^+ \vdash^{N'} t{\downarrow_{\mathcal{R}_\mathcal{E}}}$. It is easy to see that $\mathsf{E}^+ \models M \sim M'$ and $\mathsf{E}^+ \models N \sim N'$ as well. Because $(\mathsf{F}^+,\mathsf{E}^+)$ is a saturated knowledge base we apply Proposition 1 and deduce that $\mathsf{E}^+ \models M' \sim N'$, and thus $\mathsf{E}^+ \models M \sim N$, which easily implies $\mathsf{E} \models M \sim N$. □

B Proofs of Section 5

B.1 Subterm convergent equational theories

Lemma 10 *Let $\mathcal{E}$ be a subterm convergent equational theory and $\mathcal{R}_\mathcal{E}$ be its associated rewrite system. For any frame φ, and any (F,E) such that $\text{Init}(\varphi) \Longrightarrow (\mathsf{F},\mathsf{E})$, we have that:*

1. $\{\hat{\mathsf{f}} \mid \mathsf{f} \in \mathsf{F}$ *and* f *is a solved deduction fact*$\} \subseteq \mathcal{Q}(\varphi)$ *and* $\mathcal{Q}(\varphi)$ *is finite;*
2. $\mathsf{m_f}(\mathsf{f}_0) <_\mathsf{f} \mathsf{m_f}(\mathsf{f}_1)$ *where* f_0, f_1 *are defined as in rule F-Solving;*
3. $\mathsf{m_e}(\mathsf{f}_0) <_\mathsf{e} \mathsf{m_e}(\mathsf{f}_1)$ *where* f_0, f_1 *are defined as in rule E-Solving*

where $\mathcal{Q}$, $\mathsf{m_f}$, $\mathsf{m_e}$, $<_\mathsf{f}$, and $<_\mathsf{e}$ are defined w.r.t. the rewrite system $\mathcal{R}_\mathcal{E}$ as described in Section 5.2.

Proof The proof of item 1 is done by induction on the number of saturation steps needed to reach (F,E). To ease the induction we strengthen the induction hypothesis and prove a slightly stronger statement. We define $\mathcal{Q}'(\varphi,\mathsf{F})$ as the smallest set such that

1. $[t \mid \emptyset] \in \mathcal{Q}'(\varphi,\mathsf{F})$, where $t \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
2. $[f(x_1,\ldots,x_k) \mid x_1,\ldots,x_k] \in \mathcal{Q}'(\varphi,\mathsf{F})$, where $ar(f) = k$
3. $[r\sigma \mid t_1,\ldots,t_k] \in \mathcal{Q}'(\varphi,\mathsf{f})$, where:
   - $l \to r \in \mathcal{R}_\mathcal{E}$
   - $\sigma : \text{var}(l) \to \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ is a partial function
   - $l\sigma = C[t_1,\ldots,t_k]$ for some context C
   - $r\sigma \in \text{st}(D[t_1,\ldots,t_k,u_1,\ldots,u_n])$ for some public context D and some terms u_i such that $[u_i \mid \emptyset] \in \hat{\mathsf{F}}$

– $\exists i : t_i \notin \mathcal{X}$

In the following when a projection $\hat{\mathsf{f}}$ corresponds to one of the above 3 cases, we say that f is of type i ($1 \leq i \leq 3$). Note that a solved deduction fact is either of type 1 or 2. We prove that for any (F, E) such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$ we have that $\hat{\mathsf{F}} \subseteq \mathcal{Q}'(\varphi, \mathsf{F})$. We have that $\{\hat{\mathsf{f}} \mid \hat{\mathsf{f}} \in \mathcal{Q}'(\varphi, \mathsf{F}) \text{ and } \hat{\mathsf{f}} \text{ is solved}\} \subseteq \mathcal{Q}(\varphi)$ and this allows us to conclude. We prove the result by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$.

Base case. It is clear that for all deduction facts $\mathsf{f} \in \mathrm{Init}(\varphi)$ we have that $\hat{\mathsf{f}}$ is either of type 1 or type 2.

Inductive case. We assume that the result holds for (F, E), i.e. $\hat{\mathsf{F}} \subseteq \mathcal{Q}'(\varphi, \mathsf{F})$, and show that any possible application of a saturation rule preserves the result.

1. Consider a fact $\mathsf{f} \in \mathsf{F}$ of type 1, i.e. $\hat{\mathsf{f}} = [t \mid \emptyset]$. By applying rule Narrowing to it, we obtain a fact f' such that $\hat{\mathsf{f}'} = [t' \mid \emptyset]$ with $t \rightarrow_{\mathcal{R}_\mathcal{E}} t'$. As $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$, we have that $t' \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ and therefore f' is of type 1.
2. Consider a fact $\mathsf{f} \in \mathsf{F}$ of type 2, i.e. $\hat{\mathsf{f}} = [f(x_1, \dots, x_k) \mid x_1, \dots, x_k]$. As all positions of the term $f(x_1, \dots, x_k)$, except the head are variables, rule Narrowing can only be applied at this position. Let $l \rightarrow r \in \mathcal{R}_\mathcal{E}$ be the rewrite rule involved in this step. We obtain a fact f' such that $\hat{\mathsf{f}'} = [r\tau \mid x_1\tau, \dots, x_k\tau]$ where $\tau = \mathrm{mgu}(f(x_1, \dots, x_k), l)$. We distinguish two cases:
   – *Case 1: l is a variable, say x.* In such a case, $\hat{\mathsf{f}'} = [r\tau \mid x_1, \dots, x_k]$ and $r \in \mathcal{T}(\mathcal{F}, \emptyset)$. Therefore, the resulting fact f' is redundant.
   – *Case 2: l is not a variable.* In such a case, we have that $l = f(l_1, \dots, l_k)$ and $\hat{\mathsf{f}'} = [r \mid l_1, \dots, l_k]$. Let σ be such that $\mathrm{dom}(\sigma) = \emptyset$, $C = f(\_, \dots, \_)$. It is clear that $\hat{\mathsf{f}'}$ satisfies the three first conditions of a fact of type 3. Now, either $r \in \mathcal{T}(\mathcal{F}, \emptyset)$, i.e. r is a public ground term and in such a case it is clear that the fact is redundant. Otherwise, we have that r is a strict subterm of l, i.e $r \in \mathrm{st}(l_j)$ for some $1 \leq j \leq k$. Therefore the fourth condition also holds. Now, assume that all the l_i are variables (i.e. f' is solved), we show it is redundant and it is not added to the knowledge base. Indeed, in such a situation, we necessarily have that r is a variable (remember that $r \in \mathrm{st}(l_j)$) and therefore the fact f' is redundant.
3. Consider a fact $\mathsf{f} \in \mathsf{F}$ of type 3. Let $\hat{\mathsf{f}} = [r\sigma \mid t_1, \dots, t_k]$. In such a case, there exist a rewrite rule $l \rightarrow r$, a partial function $\sigma : \mathrm{var}(l) \rightarrow \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$, a context C such that $l\sigma = C[t_1, \dots, t_k]$ and we have that $r\sigma \in \mathrm{st}(D[t_1, \dots, t_k, u_1, \dots, u_n])$ for some public context D and some terms u_i such that $[u_i \mid \emptyset] \in \hat{\mathsf{F}}$. Assume that one of the side conditions of f is being solved by rule F-Solving with a solved fact $\mathsf{f}' \in \mathsf{F}$. We assume w.l.o.g. that t_1 is being solved. We distinguish two cases depending on the type of f'.
   – *Case 1:* $\hat{\mathsf{f}'} = [u_0 \mid \emptyset]$. Let $\tau = \mathrm{mgu}(u_0, t_1)$. The fact resulting from the F-Solving rule is $\mathsf{f}'' = [r\sigma\tau \mid t_2\tau, \dots, t_k\tau]$. We consider $\sigma' = \tau \cup \sigma$, $C' = C[u_0, \dots, \_]$ and $D' = D$. We can show that the first four conditions hold. If the last condition does not hold, and because the fourth holds, the resulting fact must be either of type 1 or redundant and therefore not added to the knowledge base.
   – *Case 2:* $\hat{\mathsf{f}'} = [f(x_1, \dots, x_k) \mid x_1, \dots, x_k]$. Let $\tau = \mathrm{mgu}(f(x_1, \dots, x_k), t_1)$. As t_1 is not a variable, we have that $t_1 = f(s_1, \dots, s_\ell)$. The fact resulting from the application of the rule F-Solving is $\mathsf{f}'' = [r\sigma \mid s_1, \dots, s_\ell, t_2, \dots, t_k]$. We can show that the first four conditions hold. If the last condition does not hold, and because the fourth holds, the resulting fact must be either of type 1 or redundant and therefore not added to the knowledge base.

To show items 2 and 3 it remains to be proven that $\mathsf{m_f}$ and $\mathsf{m_e}$ strictly decrease after a side condition of an unsolved fact is solved. As a side condition can only be solved by facts of type 1 or 2 this is easily shown by a case analysis. We detail the proof for $\mathsf{m_f}$. The case of $\mathsf{m_e}$ can be done in a similar way.
Let $\mathsf{f}_1 = [R \rhd t \mid X_1 \rhd t_1, \dots X_n \rhd t_n]$.

– Suppose f_1 is solved by a solved fact f_2 of type 1. Let $\hat{\mathsf{f}}_2 = [u \mid \emptyset]$ where $u \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ and $\sigma = \mathrm{mgu}(u, t_1)$. There are two possible cases. Either $u = t_1$. As $u \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ we have that u is ground and $\mathrm{dom}(\sigma) = \emptyset$. In this case $\#\mathrm{var}(t_2, \dots, t_n) = \#\mathrm{var}(t_1, \dots, t_n)$ but as $t_1 \notin \mathcal{X}$ we have that $\sum_{2 \leq i \leq n} |t_i| < \sum_{1 \leq i \leq n} |t_i|$. Or $u \neq t_1$ and $\#\mathrm{var}(t_2, \dots t_n) < \#\mathrm{var}(t_1, \dots t_n)$.

– Suppose f_1 is solved by a solved fact f_2 of type 2. Let $\hat{\mathsf{f}}_2 = [f(x_1,\dots,x_k) \mid x_1,\dots,x_k]$ and $\sigma = \mathrm{mgu}(u, t_1)$. As $t_1 \notin \mathcal{X}$ we have that $t_1 = f(s_1,\dots,s_k)$. We have that $\sigma = \{x_1 \mapsto s_1,\dots,x_k \mapsto s_k\}$ and the resulting fact f_0 is such that

$$\hat{\mathsf{f}}_0 = [t\sigma \mid \Delta] = [t\sigma \mid s_1,\dots,s_k,t_2,\dots,t_n].$$

Thus, we have that $\#\,\mathrm{var}(\Delta) = \#\,\mathrm{var}(t_1,\dots,t_n)$ and $\sum_{u\in\Delta}|u| < \sum_{1\le i\le n}|t_i|$.

This allows us to conclude the proof. □

B.2 Malleable encryption

Lemma 11 *For any frame φ, and any (F,E) such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F},\mathsf{E})$ w.r.t. $\mathcal{R}_{\mathcal{E}_{mal}}$, we have that:*

1. *$\{\hat{\mathsf{f}} \mid \mathsf{f} \in \mathsf{F}$ and f is a solved deduction fact$\} \subseteq \mathcal{Q}(\varphi)$ and $\mathcal{Q}(\varphi)$ is finite;*
2. *$\mathsf{m_f}(\mathsf{f}_0) <_\mathsf{f} \mathsf{m_f}(\mathsf{f}_1)$ where f_0, f_1 are defined as in rule F-Solving;*
3. *$\mathsf{m_e}(\mathsf{f}_0) <_\mathsf{e} \mathsf{m_e}(\mathsf{f}_1)$ where f_0, f_1 are defined as in rule E-Solving*

where $\mathcal{Q}$, $\mathsf{m_f}$, $\mathsf{m_e}$, $<_\mathsf{f}$, and $<_\mathsf{e}$ are defined w.r.t. to the rewrite system $\mathcal{R}_{\mathcal{E}_{mal}}$ as described in Section 5.2.

Proof Let $\mathcal{E} = \mathcal{E}_{mal}$. The proof of item 1 is done by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F},\mathsf{E})$. To ease the induction we strengthen the induction hypothesis and prove a slightly stronger statement. We define $\mathcal{Q}'(\varphi)$ as the smallest set such that:

1. $[t \mid \emptyset] \in \mathcal{Q}'(\varphi)$, for every $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
2. $[f(x_1,x_2) \mid x_1,x_2] \in \mathcal{Q}'(\varphi)$, where $f \in \{enc, dec, mal\}$
3. $[enc(x,t) \mid x] \in \mathcal{Q}'(\varphi)$, if there exists t' such that $enc(t',t) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
4. $[x \mid enc(x,y), y] \in \mathcal{Q}'(\varphi)$
5. $[enc(z,y) \mid enc(x,y), z] \in \mathcal{Q}'(\varphi)$
6. $[t \mid t_1,\dots,t_k] \in \mathcal{Q}'(\varphi)$, if $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ and $C[t_1,\dots,t_k] \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for some context C
7. $[x \mid x, t_1,\dots,t_k]$, where $C[t_1,\dots,t_k] \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for some context C

In the following when a projection $\hat{\mathsf{f}}$ corresponds to one of the above 7 cases, we say that f is of type i ($1 \le i \le 7$). We prove that for any (F,E) such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F},\mathsf{E})$ we have that $\hat{\mathsf{F}} \subseteq \mathcal{Q}'(\varphi)$. It is easy to see that $\{\hat{\mathsf{f}} \mid \hat{\mathsf{f}} \in \mathcal{Q}'(\varphi)$ and $\hat{\mathsf{f}}$ is solved$\} \subseteq \mathcal{Q}(\varphi)$, this will indeed allows us to conclude. We prove the result by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F},\mathsf{E})$.

Base case. It is clear that for all deduction facts $\mathsf{f} \in \mathrm{Init}(\varphi)$ we have that $\hat{\mathsf{f}}$ is either of type 1 or type 2.

Inductive case. We assume that the result holds for (F,E) and show that any possible application of a saturation rule preserves the result.

– Consider a fact $\mathsf{f} \in \mathsf{F}$ of type 1, i.e. $\hat{\mathsf{f}} = [t \mid \emptyset]$ with $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$. By applying rule Narrowing, we obtain a fact f' such that $\hat{\mathsf{f}'} = [t' \mid \emptyset]$, and $t \to_{\mathcal{R}_\mathcal{E}} t'$. As $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$, it follows that $t' \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ and therefore f' is a fact of type 1.
– Consider a fact $\mathsf{f} \in \mathsf{F}$ of type 2 such that $\hat{\mathsf{f}} = [f(x_1,x_2) \mid x_1,x_2]$. By applying the rule Narrowing we obtain a fact of type 4, or 5.
– Consider a fact $\mathsf{f} \in \mathsf{F}$ of type 3, then $\hat{\mathsf{f}} = [enc(x,t) \mid x]$ and the rule Narrowing can only be applied on a position in t. Therefore, Narrowing will produce another fact $\hat{\mathsf{f}}' = [enc(x,u) \mid x]$, where $t \to u$. As there exists t' such that $enc(t',t) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ by definition of $\mathrm{st}_{\mathcal{R}_\mathcal{E}}$, $enc(t',u) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ yielding again a fact of type 3.
– Consider a fact $\mathsf{f} \in \mathsf{F}$ of type 4, then its unsolved side condition can be solved using a fact of type 1, 2 or 3. In the first case, we obtain a fact of type 6. In the second case, we obtain a redundant fact. In the third case, we obtain a fact of type 7.
– Consider a fact $\mathsf{f} \in \mathsf{F}$ of type 5, its unsolved side condition can be solved using a fact of type 1, 2 or 3. In the first case, we obtain a fact of type 3. In the second and third case, we obtain a redundant fact.

– Consider a fact $\mathsf{f} \in \mathsf{F}$ of type 6 or 7, its unsolved side conditions can be solved using a fact of type 1, 2 or 3. Let f' be the new fact obtained by applying the F-Solving rule. If f' is unsolved, it has the same type as f. If f' is solved, it is either of type 1 if f is of type 6 or it is redundant if f is of type 7.

To show items 2 and 3 it remains to be proven that $\mathsf{m_f}$ and $\mathsf{m_e}$ strictly decrease after a side condition of an unsolved fact is solved. As side conditions can only be solved by facts of type 1-3 this is easily shown by a case analysis. We detail the proof for $\mathsf{m_f}$. The case of $\mathsf{m_e}$ can be done in a similar way.
Let $\mathsf{f}_1 = [R \rhd t \mid X_1 \rhd t_1, \ldots X_n \rhd t_n]$. The case where f_1 is solved by a fact f_2 of type 1 (resp. type 2) is similar to the proof done in Lemma 10. It remains the case where f_2 is of type 3.

Let $\hat{\mathsf{f}}_2 = [enc(x, u) \mid x]$ and $\sigma = \mathrm{mgu}(enc(x, u), t_1)$. As there exists u' such that $enc(u', u) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ we have that u is ground. As $t_1 \notin \mathcal{X}$ we have that $t_1 = enc(t_1', t_1'')$. The projection of the resulting fact f_0 is $\hat{\mathsf{f}}_0 = [t\sigma \mid x\sigma, t_2\sigma, \ldots, t_n\sigma]$. We distinguish two cases. Either $\sigma = \{x \mapsto t_1'\}$ and $\hat{\mathsf{f}}_0 = [t \mid t_1', t_2, \ldots, t_n]$. In such a case $\#\,\mathrm{var}(t_2, \ldots, t_n) \leq \#\,\mathrm{var}(t_1, \ldots, t_n)$ and $\sum_{2\leq i\leq n} |t_i| < \sum_{1\leq i\leq n} |t_i|$. Otherwise, we have that $\#\,\mathrm{var}(t_2, \ldots, t_n) < \#\,\mathrm{var}(t_1, \ldots, t_n)$. $\square$

B.3 Trap-door commitment

The following convergent equational theory $\mathcal{E}_{td}$ is a model for trap-door commitment:

1. $open(td(x, y, z), y) = x$
2. $td(x_2, f(x_1, y, z, x_2), z) = td(x_1, y, z)$
3. $open(td(x_1, y, z), f(x_1, y, z, x_2)) = x_2$
4. $f(x_2, f(x_1, y, z, x_2), z, x_3) = f(x_1, y, z, x_3)$

We will refer below to the four corresponding rewrite rules as R1, R2, R3 and R4.

Lemma 12 *For any frame φ, and any (F, E) such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$, we have that:*

1. *$\{\hat{\mathsf{f}} \mid \mathsf{f} \in \mathsf{F}$ and f is a solved deduction fact$\} \subseteq \mathcal{Q}(\varphi)$ and $\mathcal{Q}(\varphi)$ is finite;*
2. *$\mathsf{m_f}(\mathsf{f}_0) <_\mathsf{f} \mathsf{m_f}(\mathsf{f}_1)$ where f_0, f_1 are defined as in rule F-Solving;*
3. *$\mathsf{m_e}(\mathsf{f}_0) <_\mathsf{e} \mathsf{m_e}(\mathsf{f}_1)$ where f_0, f_1 are defined as in rule E-Solving*

where $\mathcal{Q}(\varphi)$ is defined as the smallest set that contains:

1. *$[t \mid \emptyset]$, for every $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$*
2. *$[td(t_1, r, tp) \mid \emptyset]$ such that $f(t_1, r, tp, t_2) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for some t_2*
3. *$[g(x_1, \ldots, x_k) \mid x_1, \ldots, x_k]$, where $g \in \{open, td, f\}$ and $ar(g) = k$*
4. *$[f(t_1, r, tp, x) \mid x]$, such that $f(t_1, r, tp, t_2) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for some t_2*

and $\mathsf{m_f}$, $\mathsf{m_e}$, $<_\mathsf{f}$, and $<_\mathsf{e}$ are defined with $\mathcal{E} = \mathcal{E}_{td}$ as described in Section 5.2.

Proof Let $\mathcal{E} = \mathcal{E}_{td}$. The proof of item 1 is done by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$. To ease the induction we strengthen the induction hypothesis and prove a slightly stronger statement. We define $\mathcal{Q}'(\varphi)$ as the smallest set that contains:

1. $[t \mid \emptyset]$, for every $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
2. $[td(t_1, r, tp) \mid \emptyset]$ such that $f(t_1, r, tp, t_2) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for some t_2
3. $[g(x_1, \ldots, x_k) \mid x_1, \ldots, x_k]$, where $g \in \{open, td, f\}$ and $ar(g) = k$
4. $[f(t_1, r, tp, x) \mid x]$, such that $f(t_1, r, tp, t_2) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for some t_2
5. $[x \mid td(x, y, z), y]$
6. $[td(x_1, y, z) \mid x_2, f(x_1, y, z, x_2), z]$
7. $[x_2 \mid td(x_1, y, z), f(x_1, y, z, x_2)]$
8. $[f(x_1, y, z, x_3) \mid x_2, f(x_1, y, z, x_2), z, x_3]$
9. $[x_2 \mid x_1, y, z, f(x_1, y, z, x_2)]$
10. $[x_2 \mid td(x, y, z), x, y, z, x_2]$
11. $[x \mid f(t_1, r, tp, x)]$ for every $t_1, r, tp \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
12. $[x \mid td(t, r, tp), x]$ for every $t, r, tp \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
13. $[x \mid x, t_1, \ldots, t_k]$ for every $t_1, \ldots, t_k \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
14. $[t \mid td(t_1, r, tp)]$ for every $t, t_1, r, tp \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
15. $[t \mid t_1, \ldots, t_k]$ for every $t, t_1, \ldots, t_k \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$, $k \geq 1$
16. $[td(t, r, tp) \mid t_1, \ldots, t_k]$, $\exists t'\ f(t, r, tp, t') \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$, $t_1, \ldots, t_k \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi), k \geq 1$
17. $[td(t, r, tp) \mid x, t_1, \ldots, t_k]$, $\exists t'\ f(t, r, tp, t') \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$, $t_1, \ldots, t_k \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi), k \geq 1$

18. $[f(t,r,tp,x) \mid x, t_1, \ldots, t_k]$, $\exists t'\ f(t,r,tp,t') \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$, $t_1, \ldots, t_k \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
19. $[f(t,r,tp,x) \mid x, x', t_1, \ldots, t_k]$, $\exists t'\ f(t,r,tp,t') \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$, $t_1, \ldots, t_k \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$

In the following when a projection $\hat{\mathsf{f}}$ corresponds to one of the above 19 cases, we say that f is of type i $(1 \leq i \leq 19)$. We prove that for any (F, E) such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$ we have that $\hat{\mathsf{F}} \subseteq \mathcal{Q}'(\varphi)$. It is easy to see that $\{\hat{\mathsf{f}} \mid \hat{\mathsf{f}} \in \mathcal{Q}'(\varphi)$ and $\hat{\mathsf{f}}$ is solved$\} \subseteq \mathcal{Q}(\varphi)$, this will indeed allows us to conclude. We prove the result by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$.

Base case. It is clear that all deduction facts $\mathsf{f} \in \mathrm{Init}(\varphi)$ are either of type 1 or type 3.

Inductive case. We assume that the result holds for (F, E) and show that any possible application of a saturation rule preserves the result. We summarize case analysis in the following two matrices.

Narrowing	R1	R2	R3	R4
type 1	1	1	1	1
type 2	2	2	2	2
type 3	5	6	7	8
type 4	4	4	4	4

F-Solving	type 1	type 2	type 3	type 4
type 5	15	15	redundant	impossible
type 6	16	impossible	redundant	17
type 7	11 or 14	11	9 or 10	12
type 8	18	impossible	redundant	19
type 9	15	impossible	redundant	13
type 10	13	13	redundant	impossible
type 11	1	impossible	13	redundant
type 12	redundant	redundant	13	impossible
type 13	13 or redundant	13 or redundant	13	13
type 14	1	1	15	impossible
type 15	15 or 1	15 or 1	15	15
type 16	16 or 2	16 or 2	16	16
type 17	17 or 2	17 or 2	17	17
type 18	18 or 4	18 or 4	18	18
type 19	19 or 4	19 or 4	19	19

Items 2 and 3 are shown as in Lemma 11. □

B.4 Blind signature

The following convergent equational theory $\mathcal{E}_{blind}$ is a model for blind signatures:

1. $unblind(blind(x,y),y) = x$
2. $unblind(sign(blind(x,y),z),y) = sign(x,z)$
3. $checksign(sign(x,y),pk(y)) = x$

We will refer below to the three corresponding rewrite rules as R1, R2 and R3.

Lemma 13 *For any frame φ, and any (F, E) such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$, we have that:*

1. $\{\hat{\mathsf{f}} \mid \mathsf{f} \in \mathsf{F}$ *and* f *is a solved deduction fact*$\} \subseteq \mathcal{Q}(\varphi)$ *and* $\mathcal{Q}(\varphi)$ *is finite;*
2. $\mathsf{m_f}(\mathsf{f}_0) <_\mathsf{f} \mathsf{m_f}(\mathsf{f}_1)$ *where* f_0, f_1 *are defined as in rule F-Solving;*
3. $\mathsf{m_e}(\mathsf{f}_0) <_\mathsf{e} \mathsf{m_e}(\mathsf{f}_1)$ *where* f_0, f_1 *are defined as in rule E-Solving*

where $\mathcal{Q}(\varphi)$ is defined as the smallest set that contains:

1. $[t \mid \emptyset]$, *for every* $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
2. $[f(x_1, \ldots, x_k) \mid x_1, \ldots, x_k]$, *where* $f \in \mathcal{F}$ *and* $ar(f) = k$
3. $[sign(t,x) \mid x]$, *for every* $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
4. $[sign(t,t') \mid \emptyset]$, *for every* $t, t' \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$

and $\mathsf{m_f}$, $\mathsf{m_e}$, $<_\mathsf{f}$, *and* $<_\mathsf{e}$ *are defined with* $\mathcal{E} = \mathcal{E}_{blind}$ *as described in Section 5.2.*

Proof Let $\mathcal{E} = \mathcal{E}_{blind}$. The proof of item 1 is done by induction on the number of saturation steps of $\text{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$. To ease the induction we strengthen the induction hypothesis and prove a slightly stronger statement. We define $\mathcal{Q}'(\varphi)$ as the smallest set that contains:

1. $[t \mid \emptyset]$, for every $t \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
2. $[f(x_1, \ldots, x_k) \mid x_1, \ldots, x_k]$, where $f \in \mathcal{F}$ and $ar(f) = k$
3. $[sign(t, x) \mid x]$, for every $t \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
4. $[sign(t, t') \mid \emptyset]$, for every $t, t' \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
5. $[x \mid blind(x, y), y]$
6. $[sign(x, z) \mid sign(blind(x, y), z), y]$
7. $[x \mid sign(x, y), pk(y)]$
8. $[sign(x, z) \mid blind(x, y), z, y]$
9. $[x \mid sign(x, y), y]$
10. $[x \mid x, y, pk(y)]$
11. $[t \mid t_1, \ldots, t_k]$ if $C[t_1, \ldots, t_k] \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for some context C and $t \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
12. $[sign(t, t') \mid t_1, \ldots, t_k]$ if $C[t_1, \ldots, t_k] \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for some context C, $k \geq 1$, and $t, t' \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
13. $[t \mid pk(t')]$, for every $t, t' \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
14. $[x \mid sign(x, t)]$, for every $t \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
15. $[t \mid y, pk(y)]$, for every $t \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
16. $[sign(t, z) \mid z, t_1, \ldots, t_k]$ if $C[t_1, \ldots, t_k] \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for some context C, $k \geq 1$, and $t \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
17. $[x \mid x, t_1, \ldots, t_k]$ if $C[t_1, \ldots, t_k] \in \text{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for some context C

In the following when a projection $\hat{\mathsf{f}}$ corresponds to one of the above 17 cases, we say that f is of type i ($1 \leq i \leq 17$). We prove that for any (F, E) such that $\text{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$ we have that $\hat{\mathsf{F}} \subseteq \mathcal{Q}'(\varphi)$. It is easy to see that $\{\hat{\mathsf{f}} \mid \hat{\mathsf{f}} \in \mathcal{Q}'(\varphi)$ and $\hat{\mathsf{f}}$ is solved$\} \subseteq \mathcal{Q}(\varphi)$, this will indeed allows us to conclude. We prove the result by induction on the number of saturation steps of $\text{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$.

Base case. It is clear that all deduction facts $\mathsf{f} \in \text{Init}(\varphi)$ are either of type 1 or type 2.

Inductive case. We assume that the result holds for (F, E) and show that any possible application of a saturation rule preserves the result. We summarize the case analysis in the following two matrices.

Narrowing	R1	R2	R3
type 1	1	1	1
type 2	5	6	7
type 3	3	3	3
type 4	4	4	4

F-Solving	type 1	type 2	type 3	type 4
type 5	11	redundant	impossible	impossible
type 6	12	8	16	12
type 7	13 or 14	9 or 10	15	13
type 8	16	redundant	impossible	impossible
type 9	11	redundant	1	11
type 10	17	redundant	impossible	impossible
type 11	11 or 1	11	11	11 or 1
type 12	12 or 4	12	12	12 or 4
type 13	1	11	impossible	impossible
type 14	1	17	11	1
type 15	11	1	impossible	impossible
type 16	16 or 3	16	16	16 or 3
type 17	17 or redundant	17	17	17 or redundant

Items 2 and 3 are shown as in Lemma 11. □

B.5 Addition

The following convergent equational theory $\mathcal{E}_{add}$ is a simple model of addition introduced in [1]:

1. $plus(x, s(y)) = plus(s(x), y)$
2. $plus(x, 0) = x$
3. $pred(s(x)) = x$

We will refer below to the three corresponding rewrite rules as R1, R2 and R3.

Lemma 14 *For any frame φ, and any (F, E) such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$, we have that:*

1. *$\{\hat{\mathsf{f}} \mid \mathsf{f} \in \mathsf{F}$ and f is a solved deduction fact$\} \subseteq \mathcal{Q}(\varphi)$ and $\mathcal{Q}(\varphi)$ is finite;*
2. *$\mathsf{m_f}(\mathsf{f}_0) <_\mathsf{f} \mathsf{m_f}(\mathsf{f}_1)$ where f_0, f_1 are defined as in rule F-Solving;*
3. *$\mathsf{m_e}(\mathsf{f}_0) <_\mathsf{e} \mathsf{m_e}(\mathsf{f}_1)$ where f_0, f_1 are defined as in rule E-Solving*

where $\mathcal{Q}(\varphi)$ is defined as the smallest set that contains:

1. *$[t \mid \emptyset]$, for every $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$*
2. *$[f(x_1, \ldots, x_k) \mid x_1, \ldots, x_k]$, where $f \in \{s, plus, pred, 0\}$ and $ar(f) = k$*
3. *$[plus(s^n(x), t) \mid x]$, if $s^n(t) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for $n \geq 0$*

and $\mathsf{m_f}$, $\mathsf{m_e}$, $<_\mathsf{f}$, and $<_\mathsf{e}$ are defined with $\mathcal{E} = \mathcal{E}_{add}$ as described in Section 5.2.

Proof Let $\mathcal{E} = \mathcal{E}_{add}$. The proof of item 1 is done by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$. To ease the induction we strengthen the induction hypothesis and prove a slightly stronger statement. We define $\mathcal{Q}'(\varphi)$ as the smallest set that contains:

1. $[t \mid \emptyset]$, for every $t \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$
2. $[f(x_1, \ldots, x_k) \mid x_1, \ldots, x_k]$, where $f \in \mathcal{F}$ and $ar(f) = k$
3. $[plus(s^n(x), t) \mid x]$, if $s^n(t) \in \mathrm{st}_{\mathcal{R}_\mathcal{E}}(\varphi)$ for $n \geq 0$
4. $[x \mid x, 0]$
5. $[plus(s(x), y) \mid x, s(y)]$
6. $[x \mid s(x)]$

In the following when a projection $\hat{\mathsf{f}}$ corresponds to one of the above 6 cases, we say that f is of type i ($1 \leq i \leq 6$). We prove that for any (F, E) such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$ we have that $\hat{\mathsf{F}} \subseteq \mathcal{Q}'(\varphi)$. It is easy to see that $\{\hat{\mathsf{f}} \mid \hat{\mathsf{f}} \in \mathcal{Q}'(\varphi)$ and $\hat{\mathsf{f}}$ is solved$\} \subseteq \mathcal{Q}(\varphi)$, this will indeed allows us to conclude. We prove the result by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$.

Base case. It is clear that all deduction facts $\mathsf{f} \in \mathrm{Init}(\varphi)$ are either of type 1 or type 2.

Inductive case. We assume that the result holds for (F, E) and show that any possible application of a saturation rule preserves the result. We summarize the case analysis in the following two matrices.

Narrowing	R1	R2	R3
type 1	1	1	1
type 2	5	4	6
type 3	3	redundant or 3	3

F-Solving	type 1	type 2	type 3
type 4	redundant	redundant	impossible
type 5	3	redundant	impossible
type 6	1	redundant	impossible

To show item 2 and 3, it remains to be proven that $\mathsf{m_f}$ and $\mathsf{m_e}$ strictly decrease after a side condition of an unsolved fact is solved. A side condition can only be solved by facts of type 1, 2 or 3. We show the result by a case analysis.
Let $\mathsf{f}_1 = [R \rhd t \mid X_1 \rhd t_1, \ldots, X_n \rhd t_n]$.

- If the solved fact is of type 1 or 2, the proof is similar to the reasoning done in Lemma 10.
- It is easy to see that a solved fact of type 3 cannot be used to solved a side condition of an unsolved fact (types 4-6). Indeed, the side conditions which are are not variables, are either 0 or a term of the form $s(x)$ and hence unification is impossible.

Let $\mathsf{f} = [U \sim V \mid X_1 \rhd t_1, \ldots, X_n \rhd t_n]$

- If the solved fact is of type 1 or 2, the proof is similar to the reasoning done in Lemma 10.
- A solved fact of type 3 can be used to solve a side condition of the form $X \rhd t$ when t is headed with the symbol *plus*. It is easy to see (since we already know the form of the deduction facts) that the only terms t occurring in a side condition of an equational fact and headed with *plus* are ground. This allows us to conclude that the measure $\mathsf{m_e}$ decreases also in this case. □

B.6 Homomorphic encryption

Lemma 15 *If the saturation strategy is fair the saturation process terminates for the equational theory $\mathcal{E}_{\mathsf{hom}}$.*

Proof In the following let $\mathcal{E} = \mathcal{E}_{\mathsf{hom}}$. Orienting the five equations in $\mathcal{E}_{\mathsf{hom}}$ we obtain the following rewriting rules:

R1 $fst(pair(x, y)) \to x$
R2 $snd(pair(x, y)) \to y$
R3 $dec(enc(x, y), y) \to x$
R4 $enc(pair(x, y), z) \to pair(enc(x, z), enc(y, z))$
R5 $dec(pair(x, y), z) \to pair(dec(x, z), dec(y, z))$

For the purpose of this proof we extend the notion of extended subterm and define $\mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(t)$ to be the smallest set such that:

1. $t \in \mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(t)$,
2. $f(t_1, \ldots, t_k) \in \mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(t)$ implies $t_1, \ldots, t_k \in \mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(t)$,
3. $t' \in \mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(t)$ and $t' \to_{\mathcal{R}_\mathcal{E}} t''$ implies $t'' \in \mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(t)$.
4. $\mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(f(t_1, \ldots, t_k)) \in \mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(t)$ implies $\mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(f(s_1, \ldots, s_k)) \in \mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(t)$ for every $s_i \in \mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(t_i)$ and for every $f \in \mathcal{F}$ of arity k.

Let φ be the frame being saturated. We first show that for all knowledge bases (F, E) such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$ we have that each $\hat{\mathsf{f}} \in \hat{\mathsf{F}}$ has one of the following forms:

1. $[t \mid \emptyset]$, for some $t \in \mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(\varphi)$
2. $[fst(x) \mid x]$
3. $[snd(x) \mid x]$
4. $[enc(x, y) \mid x, y]$
5. $[dec(x, y) \mid x, y]$
6. $[pair(x, y) \mid x, y]$
7. $[C[t_1, \ldots, t_k] \mid \mathrm{var}(C)]$ where:
   – C is obtained by arbitrarily nesting the following (classes of) contexts: $C_1 = enc(\_, z_i)$, $C_2 = dec(\_, z_i)$ and $C_3 = pair(\_, \_)$, where z_i are variables.
   – C contains at least one variable.
   – $C'[t_1, \ldots, t_k] \in \mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(\phi)$, where C' is obtain from C by replacing $enc(\_, z_i)$ and $dec(\_, z_i)$ with $\_$.
8. $[x \mid pair(x, y)]$
9. $[y \mid pair(x, y)]$
10. $[x \mid enc(x, y), y]$
11. $[pair(enc(x, z), enc(y, z)) \mid pair(x, y), z]$
12. $[pair(dec(x, z), dec(y, z)) \mid pair(x, y), z]$
13. $[t \mid t_1, \ldots, t_k]$, for some $t, t_1, \ldots, t_k \in \mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(\varphi)$
14. $[C[t_1, \ldots, t_k] \mid s_1, \ldots, s_l, \mathrm{var}(C)]$ where:
   – C is obtained by arbitrarily nesting the following (classes of) contexts: $C_1 = enc(\_, z_i)$, $C_2 = dec(\_, z_i)$, and $C_3 = pair(\_, \_)$, where z_i are variables.
   – $C'[t_1, \ldots, t_k] \in \mathrm{st}^+_{\mathcal{R}_\mathcal{E}}(\phi)$, where C' is obtain from C by replacing $enc(\_, z_i)$ and $dec(\_, z_i)$ with $\_$.
   – s_i are ground terms

We show this by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}, \mathsf{E})$. In the following when a projection $\hat{\mathsf{f}}$ corresponds to one of the above 14 cases, we say that f is of type i $(1 \leq i \leq 14)$.
Base case. It is easy to see that all $\mathsf{f} \in \mathrm{Init}(\varphi)$ are indeed of type $1-6$.
Inductive case. We assume that the result holds for (F, E) and show that any possible application of a saturation rule preserves the result. We summarize case analysis in the following two matrices.

Narrowing	R1	R2	R3	R4	R5
type 1	1	1	1	1	1
type 2	8	impossible	impossible	impossible	impossible
type 3	impossible	9	impossible	impossible	impossible
type 4	impossible	impossible	impossible	11	impossible
type 5	impossible	impossible	10	impossible	12
type 6	impossible	impossible	impossible	impossible	impossible
type 7	7	7	$1, 7, 13, 14$	7	7

F-Solving	type 1	type 2	type 3	type 4	type 5	type 6	type 7
type 8	1	imp.	imp.	imp.	imp.	redundant	$7, 1$
type 9	1	imp.	imp.	imp.	imp.	redundant	$7, 1$
type 10	13	imp.	imp.	imp.	redundant	imp.	$7, 1$
type 11	7	imp.	imp.	imp.	imp.	redundant	7
type 12	7	imp.	imp.	imp.	imp.	redundant	7
type 13	$1, 13$	13	13	13	13	13	13
type 14	$7, 14$	14	14	14	14	14	14

We next show that because the strategy is fair at a given saturation step, no more facts of type 7 are added.

Lemma 16 *Suppose that the saturation strategy is fair and let*

$$\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F}_0, \mathsf{E}_0) \Longrightarrow \ldots \Longrightarrow (\mathsf{F}_i, \mathsf{E}_i) \Longrightarrow \ldots$$

be a sequence of saturation steps. If $\hat{\mathsf{f}} = [C[t_1, \ldots, t_k] \mid s_1, \ldots, s_l, \mathrm{var}(C)] \in \hat{\mathsf{F}}_0$ *is of type 7 or type 14 and* $\mathsf{F}_0 \vdash s_j$ *for all* j*, then there exists* n *such that* $\mathsf{F}_n \vdash t_i$ *for all* i.

Proof The proof is done by induction on the number of saturation steps of $\mathrm{Init}(\varphi) \Rightarrow^* (\mathsf{F}_0, \mathsf{E}_0)$.
Base case. As $\mathrm{Init}(\varphi)$ does not contain any facts of type 7 or 14 we conclude.
Inductive case. We suppose that the result holds for $(\mathsf{F}_0, \mathsf{E}_0)$ and verify that it is maintained by any possible rules that add a fact of type 7 or 14.

- Suppose we add a fact of type 7 by using rule Narrowing on a fact of type 7 in F_0 and R1 or R2. The rewriting must occur at a position in one of the t_i which is rewritten to t'_i. By induction hypothesis we have that there exists n, such that $\mathsf{F}_n \vdash t_i$. We can adapt the proof of Proposition 2 to show that because of fairness (rather than saturation) narrowing must be applied such that there exists n' such that $\mathsf{F}_{n'} \vdash t'_i$.
- Suppose we add a fact of type 7 by using rule Narrowing on a fact of type 7 in F_0 and R3. If narrowing is applied on one of the t_i the case is similar to the previous one. If narrowing is applied inside the context such that the t_i do not change we conclude by induction hypothesis.
- Suppose we add a fact of type 14 by using rule Narrowing on a fact of type 7 in F_0 and R3. Narrowing must have changed both the context and one of the t_i. Suppose w.l.o.g. $i = 1$. It must be that be that $t_1 = enc(t'_1, t''_1)$. We have to show that there exists n such that if $\mathsf{F}_n \vdash t''_1$ then $\mathsf{F}_n \vdash t'_1$ and $\mathsf{F}_n \vdash t_i$ for $2 \leq i \leq k$. $\mathsf{F}_n \vdash t_i$ is obtained by induction hypothesis. If $\mathsf{F}_n \vdash t''_1$ and because $\mathsf{F}_n \vdash enc(t'_1, t''_1)$ we can apply Narrowing such that $\mathsf{F}_{n'} \vdash t'_1$ for some n'.
- Suppose we add a fact of type 7 by using rule Narrowing on a fact of type 7 in F_0 and R4. If narrowing is applied on one of the t_i the case is similar to previous cases. If narrowing is applied inside the context such that the t_i do not change we conclude by induction hypothesis. Suppose both the context and one of the t_i change. We suppose w.l.o.g. that $i = 1$. It must be that $t_1 = pair(t'_1, t''_1)$. By induction hypothesis we have that there exists n such that $\mathsf{F}_n \vdash t_i$ for $2 \leq i \leq k$. We need to show that there exists F_n. As $\mathsf{F}_n \vdash pair(t'_1, t''_1)$ we also have that $\mathsf{F}_n \vdash fst(pair(t'_1, t''_1))$ and $\mathsf{F}_n \vdash snd(pair(t'_1, t''_1))$. Because of fairness Narrowing can be applied such that $\mathsf{F}_{n'} \vdash t'_1$ and $\mathsf{F}_{n'} \vdash t''_1$ for some n''.
- Suppose we add a fact of type 7 by using rule F-Solving on facts of type 11 and 1 in F_0. Let $pair(t_1, t_2)$ be the fact of type 1. As the strategy is fair we will add facts $[x|pair(x, y)]$ and $[y|pair(x, y)]$ by applying rule Narrowing on type 2/R1 and type 3/R2. Again by fairness we will apply solving on $pair(t_1, t_2)$ and $[x|pair(x, y)]$ as well as $[y|pair(x, y)]$. Therefore t_1 and t_2 will be generated.

- Suppose we add a fact of type 7 by using rule F-Solving on facts of type 12 and 1 in F_0. This case is similar to the previous one.
- Suppose we add a fact of type 7 by applying rule F-Solving on facts of type 8-12 with a fact of type 7 in F_0. The resulting fact is a context on the same (or a subset of the) terms t_i $(1 \leq i \leq k)$ as the initial type 7 fact. We conclude by induction hypothesis.
- Suppose we add a fact of type 7 by applying rule F-Solving on a fact of type 14 with a fact of type 1 in F_0. The type 14 fact has only one ground side condition s_1 which is solved by the type 1 fact. Hence $[s_1] \in \hat{\mathsf{F}}_0$ and $\mathsf{F}_0 \vdash s_1$. We can apply the induction hypothesis and conclude.
- Suppose we add a fact of type 14 by applying rule F-Solving on a fact of type 14 with a fact of type i $(1 \leq i \leq 14)$ in F_0. We directly conclude by induction hypothesis. $\square$

There are a finite number of solved facts other than of type 7. There exist only a finite number of t_i which can occur in facts of type 7 as they are in $\mathrm{st}^{+}_{\mathcal{R}_{\mathcal{E}}}(\varphi)$.

Hence it follows from Lemma 16 that for any fair saturation sequence, at some moment all new facts of type 7 become redundant and therefore are not added to the knowledge base. Therefore any fair saturation sequence only contains a finite number of solved facts.

We know that after some number n of saturation steps, no more solved deduction facts are added to the knowledge base. We now show that a finite number of unsolved facts are added after this stage. Indeed, after n iterations, as no more solved facts are added to the knowledge base, the only types of facts potentially added are 13 and 14. The side conditions of these facts contain only ground terms or variables. By solving one of the ground side conditions the cardinality of the side condition decreases ensuring termination.

We now show that all equational facts are of the form $[M \sim N \mid X_1 \rhd t_1, \ldots, X_k \rhd t_k]$, for some M, N where either $t_i \in \mathcal{X}$ or $t_i = C[s_1, \ldots, s_l]$ for some ground terms s_j $(1 \leq j \leq l)$ and for some context C obtained by arbitrary nesting of contexts $C_1 = enc(\_, z_n)$, $C_2 = dec(\_, z_n)$, $C_3 = pair(\_, \_)$ and $C_4 = \_$, where z_n are variables.

This is true for the equational facts obtained by rule Unifying. When applying rule E-Solving on a side condition of the above type we consider the following cases:

- if we solve $X_i \rhd t_i$ with a type 1 fact, we easily conclude;
- if we solve $X_i \rhd t_i$ with a fact of type 2, 3, 4, 5, 6, the result is immediate;
- if we solve $X_i \rhd t_i$ (where $t_i = C[s_1, \ldots, s_l]$) with a type 7 fact $[C'[u_1, \ldots, u_m] \mid \mathrm{var}(C')]$, we note that $\mathrm{mgu}(t_i, C'[u_1, \ldots, u_m])$ is such that variables are mapped to either variables or ground terms. Therefore the property holds.

Using again the measure

$$\mathsf{m_e}([M \sim N \mid X_1 \rhd t_1, \ldots, X_k \rhd t_k]) = (\#\,\mathrm{var}(t_1, \ldots, t_k), |t_1| + \ldots + |t_k|)$$

and the lexicographic order $<_{\mathsf{e}}$ on pairs, we obtain that $\mathsf{f}_0 <_{\mathsf{e}} \mathsf{f}_1$ for all f_0 and f_1 as in rule F-Solving.

Automating security analysis: symbolic equivalence of constraint systems [⋆]

Vincent Cheval, Hubert Comon-Lundh, and Stéphanie Delaune

LSV, ENS Cachan & CNRS & INRIA Saclay Île-de-France

Abstract. We consider security properties of cryptographic protocols, that are either trace properties (such as confidentiality or authenticity) or equivalence properties (such as anonymity or strong secrecy).
Infinite sets of possible traces are symbolically represented using *deducibility constraints*. We give a new algorithm that decides the trace equivalence for the traces that are represented using such constraints, in the case of signatures, symmetric and asymmetric encryptions. Our algorithm is implemented and performs well on typical benchmarks. This is the first implemented algorithm, deciding symbolic trace equivalence.

1 Introduction

Security protocols are small distributed programs aiming at some security goal, though relying on untrusted communication media. Formally proving that such a protocol satisfies a security property (or finding an attack) is an important issue, in view of the economical and social impact of a failure.

Starting in the 90s, several models and automated verification tools have been designed. For instance both protocols, intruder capabilities and security properties can be formalized within first-order logic and dedicated resolution strategies yield relevant verification methods [18, 21, 6]. Another approach, initiated in [19], consists in symbolically representing the traces using deducibility constraints. Both approaches were quite successful in finding attacks/proving security protocols. There are however open issues, that concern the extensions of the methods to larger classes of protocols/properties [11]. For instance, most efforts and successes only concerned, until recently, *trace properties*, i.e., security properties that can be checked on each individual sequence of messages corresponding to an execution of the protocol. A typical example of a trace property is the *confidentiality*, also called *weak secrecy*: a given message m should not be deducible from any sequence of messages, that corresponds to an execution of the protocol. Agreement properties, also called *authenticity properties*, are other examples of trace properties.

There are however security properties that cannot be stated as properties of a single trace. Consider for instance a voter casting her vote, encrypted with a public key of a server. Since there are only a fixed, known, number of possible

⋆ This work has been partially supported by the ANR project SeSur AVOTÉ.

plaintexts, the confidentiality is not an issue. A more relevant property is the ability to relate the voter's identity with the plaintext of the message. This is a property in the family of *privacy* (or *anonymity*) properties [15]. Another example is the *strong secrecy*: m is strongly secret if replacing m with any m' in the protocol, would yield another protocol that is indistinguishable from the first one: not only m itself cannot be deduced, but the protocol also does not leak any piece of m. These two examples are not trace properties, but *equivalence properties*: they can be stated as the indistinguishability of two processes. In the present paper, we are interested in automating the proofs of equivalence properties. As far as we know, there are only three series of works that consider the automation of equivalence properties for security protocols[1].

The first one [7] is an extension of the first-order logic encoding of the protocols and security properties. The idea is to overlap the two processes that are supposedly equivalent, forming a *bi-process*, then formalize in first-order logic the simultaneous moves (the single move of the bi-process) upon reception of a message. This method checks a stronger equivalence than observational equivalence, hence it fails on some simple (cook up) examples of processes that are equivalent, but their overlapping cannot be simulated by the moves of a single bi-process. The procedure might also not terminate or produce false attacks, but considers an unbounded number of protocol instances.

The second one [3] (and [14]) assumes a fixed (bounded) number of sessions. Because of the infinite number of possible messages forged by an attacker, the number of possible traces is still infinite. The possible traces of the two processes are symbolically represented by two deducibility constraints. Then [3] provides with a decision procedure, roughly checking that the solutions, *and the recipes that yield the solutions* are identical for both constraints. This forces to compute the solutions and the associated recipes and yields an unpractical algorithm.

The third one [17, 9] is based on an extension of the small attack property of [20]. They show that, if two processes are not equivalent, then there must exist a small witness of non-equivalence. A decision of equivalence can be derived by checking every possible small witness. As in the previous method, the main problem is the practicality. The number of small witnesses is very large as all terms of size smaller than a given bound have to be considered. Consequently, neither this method nor the previous one have been implemented.

We propose in this paper another algorithm for deciding equivalence properties. As in [3, 9], we consider *trace equivalence*, which coincides with observational equivalence for determinate processes [14]. In that case, the equivalence problem can be reduced to the symbolic equivalence of finitely many pairs of deducibility constraints, each of which represents a set of traces (see [14]). We consider signatures, pairing, symmetric and asymmetric encryptions, which is slightly less general than [3, 9], who consider arbitrary subterm-convergent theories. The main idea of our method is to simultaneously solve pairs of constraints, instead of solving each constraint separately and comparing the solutions, as

[1] [16] gives a logical characterization of the equivalence properties. It is not clear if this can be of any help in deriving automated decision procedures.

in [3]. These pairs are successively split into several pairs of systems, while preserving the symbolic equivalence: roughly, the father pair is in the relation if, and only if, all the sons pairs are in the relation. This is not fully correct, since, for termination purposes, we need to keep track of some earlier splitting, using additional predicates. Such predicates, together with the constraint systems, yield another notion of equivalence, which is preserved upwards, while the former is preserved downwards. When a pair of constraints cannot be split any more, then the equivalence can be trivially checked.

A preliminary version of the algorithm has been implemented and works well (within a few seconds) on all benchmarks. The same implementation can also be used for checking the static equivalence and for checking the constraints satisfiability. We also believe that it is easier (w.r.t. [3, 9]) to extend the algorithm to a more general class of processes (including disequality tests for instance) and to avoid the detour through trace equivalence. This is needed to go beyond the class of determinate processes.

We first state precisely the problem in Section 2, then we give the algorithm, actually the transformation rules, in Section 3. We sketch the correctness and termination proofs in Section 4 and provide with a short summary of the experiments in Section 5. Detailed proofs of the results can be found in [8].

2 Equivalence properties and deducibility constraints

We use the following straightfoward example for illustrating some definitions:

Example 1. Consider the following very simple handshake protocol:

$$\begin{array}{l} A \to B : \mathsf{enc}(N_A, K_{AB}) \\ B \to A : \mathsf{enc}(f(N_A), K_{AB}) \end{array}$$

The agent A sends a random message N_A to B, encrypted with a key K_{AB}, that is shared by A and B only. The agent B replies by sending $f(N_A)$ encrypted with the same key. The function f is any function, for instance a hash function.

Consider only one session of this protocol: a sends $\mathsf{enc}(n_a, k_{ab})$ and waits for $\mathsf{enc}(f(n_a), k_{ab})$. The agent b is expecting a message of the form $\mathsf{enc}(x, k_{ab})$. The variable x represents the fact that b does not know in advance what is this randomly generated message. Then he replies by sending out $\mathsf{enc}(f(x\sigma), k_{ab})$. All possible executions are obtained by replacing x with any message $x\sigma$ such that the attacker can supply with $\mathsf{enc}(x\sigma, k_{ab})$ and then with $\mathsf{enc}(f(n_a), k_{ab})$. This is represented by the following constraint:

$$C := \begin{cases} \quad a,\ b,\ \mathsf{enc}(n_a, k_{ab}) \overset{?}{\vdash} \mathsf{enc}(x, k_{ab}) \\ a,\ b,\ \mathsf{enc}(n_a, k_{ab}),\ \mathsf{enc}(f(x), k_{ab}) \overset{?}{\vdash} \mathsf{enc}(f(n_a), k_{ab}) \end{cases}$$

Actually, C has only one solution: x has to be replaced by n_a. There is no other way for the attacker to forge a message of the form $\mathsf{enc}(x, k_{ab})$.

2.1 Function symbols and terms

We will use the set of function symbols $\mathcal{F} = \mathcal{N} \cup \mathcal{C} \cup \mathcal{D}$ where:

- $\mathcal{C} = \{\mathsf{enc},\ \mathsf{aenc},\ \mathsf{pub},\ \mathsf{sign},\ \mathsf{vk},\ \langle\ \rangle\}$ is the set of *constructors*;
- $\mathcal{D} = \{\mathsf{dec},\ \mathsf{adec},\ \mathsf{check},\ \mathsf{proj}_1,\ \mathsf{proj}_2\}$ is the set of *destructors*;
- $\mathcal{N}$ is a set of constants, called *names*.

In addition, $\mathcal{X}$ is a set of variables x, y, z,... The *constructor terms* (resp. *ground constructor terms*) are built on $\mathcal{C}$, $\mathcal{N}$ and $\mathcal{X}$ (resp. $\mathcal{C},\mathcal{N}$). The term rewriting system below is convergent: we let $t\!\downarrow$ be the normal form of t.

$$\begin{array}{lll} \mathsf{adec}(\mathsf{aenc}(x,\mathsf{pub}(y)),y) \to x & \mathsf{proj}_1(\langle x,y\rangle) \to x & \mathsf{dec}(\mathsf{enc}(x,y),y) \to x \\ \mathsf{check}(\mathsf{sign}(x,y),\mathsf{vk}(y)) \to x & \mathsf{proj}_2(\langle x,y\rangle) \to y & \end{array}$$

A (ground) *recipe* records the attacker's computation. It is used as a witness of how some deduction has been performed. Formally, it is a term built on $\mathcal{C},\mathcal{D}$ and a set of special variables $\mathcal{AX} = \{ax_1,\ldots,ax_n,\ldots\}$, that can be seen as pointers to the hypotheses, or known messages. Names are excluded from recipes: names that are known to the attacker must be given explicitly as hypotheses.

Example 2. Given $\mathsf{enc}(a,b)$ and b, the recipe $\zeta = \mathsf{dec}(ax_1,ax_2)$ is a witness of how to deduce a: $\zeta\{ax_1 \mapsto \mathsf{enc}(a,b);\, ax_2 \mapsto b\}\!\downarrow = a$.

The recipes are generalized, including possibly variables that range over recipes: (general) recipes are terms built on $\mathcal{C},\mathcal{D},\mathcal{AX}$ and $\mathcal{X}_r$, a set of recipe variables, that are written using capital letters $X, X_1, X_2, \ldots$.

We denote by $var(u)$ is the set of variables of any kind that occur in u.

2.2 Frames

The *frame* records the messages that have been sent by the participants of the protocol; it is a symbolic representation of a set of sequences of messages. The frame is also extended to record some additional informations on attacker's deductions. Typically $\mathsf{dec}(X,\zeta), i \rhd u$ records that, using a decryption with the recipe ζ, on top of a recipe X, allows to get u (at stage i). After recording this information in the frame, we may forbid the attacker to use a decryption on top of X, forcing him to use this "direct access" from the frame.

Definition 1. *A* frame ϕ *is a sequence* $\zeta_1, i_1 \rhd u_1, \ldots, \zeta_n, i_n \rhd u_n$ *where* $u_1,\ldots,u_n$ *are constructor terms,* $i_1,\ldots,i_n \in \mathbb{N}$*, and* $\zeta_1,\ldots,\zeta_n$ *are general recipes. The* domain *of the frame* ϕ*, denoted* $dom(\phi)$*, is the set* $\{\zeta_1,\ldots,\zeta_n\} \cap \mathcal{AX}$*. It must be equal to* $\{ax_1,\ldots,ax_m\}$ *for some* m *that is called the* size *of* ϕ*. A frame is* closed *when* $u_1,\ldots,u_n$ *are ground terms and* $\zeta_1,\ldots,\zeta_n$ *are ground recipes.*

Example 3. The messages of Example 1 are recorded in a frame of size 4.

$$\{ax_1, 1 \rhd a,\ ax_2, 2 \rhd b,\ ax_3, 3 \rhd \mathsf{enc}(n_a,k_{ab}),\ ax_4, 4 \rhd \mathsf{enc}(f(x),k_{ab})\}.$$

A frame ϕ defines a substitution $\{ax \mapsto u \mid ax \in dom(\phi), ax \rhd u \in \phi\}$. A closed frame is *consistent* if, for every $\zeta \rhd u \in \phi$, we have that $\zeta\phi\!\downarrow = u$.

2.3 Deducibility constraints

The following definitions are consistent with [12]. We generalize however the usual definition, including equations between recipes, for example, in order to keep track of some choices in our algorithm.

Definition 2. *A* deducibility constraint *(sometimes called simply* constraint *in what follows) is either* $\perp$ *or consists of:*

1. *a subset* S *of* $\mathcal{X}$ *(the free variables of the constraint);*
2. *a frame* ϕ*, whose size is some* m*;*
3. *a sequence* $X_1, i_1 \overset{?}{\vdash} u_1;\ \ldots;\ X_n, i_n \overset{?}{\vdash} u_n$ *where*
   - $X_1, \ldots, X_n$ *are distinct variables in* $\mathcal{X}_r$*,* $u_1, \ldots, u_n$ *are constructor terms, and* $0 \leq i_1 \leq \ldots \leq i_n \leq m$*.*
   - *for every* $0 \leq k \leq m$*,* $var(ax_k\phi) \subseteq \bigcup_{i_j < k} var(u_j)$*;*
4. *a conjunction* E *of equations and disequations between terms;*
5. *a conjunction* E' *of equations and disequations between recipes.*

The variables X_i represent the recipes that might be used to deduce the right hand side of the deducibility constraint. The indices indicate which initial segment of the frame can be used. We use this indirect representation, instead of the seemingly simpler notation of Example 1, because the transformation rules that will change the frame don't need then to be reproduced on all relevant left sides of deducibility constraints.

Example 4. Back to Example 1, the deducibility constraint is formally given by $S = \{x, y\}$, $E = E' = \emptyset$, the frame ϕ as in Example 3 and the sequence:

$$D \ = \ X_1, 3 \overset{?}{\vdash} \mathsf{enc}(x, k_{ab});\ \ X_2, 4 \overset{?}{\vdash} \mathsf{enc}(f(n_a), k_{ab}).$$

For sake of simplicity, in what follows, we will forget about the first component (the free variables). This is justified by an invariant of our transformation rules: initially all variables are free and each time new variables are introduced, their assignment is determined by an assignment of the free variables.

Definition 3. *A* solution *of a deducibility constraint* $C = (\phi, D, E, E')$ *consists of a mapping* σ *from variables to ground constructor terms and a substitution* θ *mapping* $\mathcal{X}_r$ *to ground recipes, such that:*

- *for every* $\zeta, i \rhd u \in \phi$*,* $var(\zeta\theta) \subseteq \{ax_1, \ldots, ax_i\}$ *and* $\zeta\theta(\phi\sigma)\!\downarrow = u\sigma\!\downarrow$ *(i.e. the frame is consistent after instanciating the variables);*
- *for every* $X_i, j \overset{?}{\vdash} u_i$ *in* D*,* $var(X_i\theta) \subseteq \{ax_1, \ldots, ax_j\}$ *and* $X_i\theta(\phi\sigma)\!\downarrow = u_i\sigma\!\downarrow$*;*
- *for every equation* $u \overset{?}{=} v$ *(resp.* $u \overset{?}{\neq} v$*) in* E*,* $u\sigma\!\downarrow = v\sigma\!\downarrow$ *(resp.* $u\sigma\!\downarrow \neq v\sigma\!\downarrow$*);*
- *for every equation* $\zeta \overset{?}{=} \zeta'$ *(resp.* $\zeta \overset{?}{\neq} \zeta'$*) in* E'*,* $\zeta\theta = \zeta'\theta$ *(resp.* $\zeta\theta \neq \zeta'\theta$*).*

$\mathsf{Sol}(C)$ *is the set of solutions of* C*. By convention,* $\mathsf{Sol}(\perp) = \emptyset$*.*

Example 5. Coming back to Example 4, a solution is (σ, θ) with:

- $\sigma = \{x \mapsto n_a,\ y \mapsto \langle a, \mathsf{enc}(n_a, k_{ab})\rangle\}$, and
- $\theta = \{X_1 \mapsto ax_3,\ X_2 \mapsto ax_4,\ X_3 \mapsto \langle ax_1, ax_3\rangle\}$.

Each solution of a constraint corresponds to a possible execution of the protocol, together with the attacker's actions that yield this execution. For instance an attack on the confidentiality of a term s can be modeled by adding $X, m \overset{?}{\vdash} s$ to the constraint system (X is a fresh variable and m is the size of the frame). This represents the derivability of s from the messages sent so far. Note that there might be several attacker's recipes yielding the same trace.

Example 6. Consider another very simple example: the Encrypted Password Transmission protocol [13], which is informally described by the rules:

$$\begin{array}{l} A \to B : \langle N_A, \mathsf{pub}(K_A)\rangle \\ B \to A : \mathsf{aenc}(\langle N_A, P\rangle, \mathsf{pub}(K_A)) \end{array}$$

Assume that a first sends a message whereas b is waiting for a message of the form $\langle x, \mathsf{pub}(k_a)\rangle$. Then b responds by sending $\mathsf{aenc}(\langle x, p\rangle, \mathsf{pub}(k_a))$. The corresponding deducibility constraint is (S, ϕ, D, E, E') where $S = \{x, y\}$, $E = E' = \emptyset$, and the sequences ϕ and D are as follows:

$$\phi = \begin{cases} ax_1, 1 \rhd \mathsf{pub}(k_a);\ ax_2, 2 \rhd \mathsf{pub}(k_b); \\ ax_3, 3 \rhd \langle n_a, \mathsf{pub}(k_a)\rangle; \\ ax_4, 4 \rhd \mathsf{aenc}(\langle x, p\rangle, \mathsf{pub}(k_a)) \end{cases} \qquad D = \begin{cases} X_1, 3 \overset{?}{\vdash} \langle x, \mathsf{pub}(k_a)\rangle \\ X_2, 4 \overset{?}{\vdash} \mathsf{aenc}(\langle n_a, y\rangle, \mathsf{pub}(k_a)) \end{cases}$$

There are several solutions. For instance, the "honest solution" (σ_h, θ_h) is given by $\sigma_h = \{x \mapsto n_a, y \mapsto p\}$ and $\theta_h = \{X_1 \mapsto ax_3, X_2 \mapsto ax_4\}$. Another solution is (σ, θ) where $\sigma = \{x \mapsto \mathsf{pub}(k_a), y \mapsto n_a\}$ and $\theta = \{X_1 \mapsto \langle ax_1, ax_1\rangle, X_2 \mapsto \mathsf{aenc}(\langle \mathsf{proj}_1(ax_3), \mathsf{proj}_1(ax_3)\rangle, ax_1)\}$.

2.4 Static equivalence

Two sequences of terms are *statically equivalent* if, whatever an attacker observes on the first sequence, the same observation holds on the second sequence [2]:

Definition 4. *Two closed frames ϕ and ϕ' having the same size m are* statically equivalent*, which we write $\phi \sim_s \phi'$, if*

1. *for any ground recipe ζ such that $var(\zeta) \subseteq \{ax_1, \ldots, ax_m\}$, we have that*
$$\zeta\phi\!\downarrow \textit{ is a constructor term if, and only if, } \zeta\phi'\!\downarrow \textit{ is a constructor term}$$
2. *for any ground recipes ζ, ζ' such that $var(\{\zeta, \zeta'\}) \subseteq \{ax_1, \ldots, ax_m\}$, and the terms $\zeta\phi\!\downarrow$, $\zeta'\phi\!\downarrow$ are constructor terms, we have that*
$$\zeta\phi\!\downarrow = \zeta'\phi\!\downarrow \textit{ if, and only, if } \zeta\phi'\!\downarrow = \zeta'\phi'\!\downarrow.$$

Example 7. Consider the frames $\phi_1 = \{ax_1 \rhd a, ax_2 \rhd \mathsf{enc}(a, b), ax_3 \rhd b\}$ and $\phi_2 = \{ax_1 \rhd a, ax_2 \rhd \mathsf{enc}(c, b), ax_3 \rhd b\}$. $\phi_1 \not\sim_s \phi_2$ since choosing $\zeta = \mathsf{dec}(ax_2, ax_3)$ and $\zeta' = ax_1$ yields $\zeta\phi_1\!\downarrow = \zeta'\phi_1\!\downarrow = a$ while $\zeta\phi_2\!\downarrow \neq \zeta'\phi_2\!\downarrow$.

On the other hand, $\{ax_1 \rhd a, ax_2 \rhd \mathsf{enc}(a, b)\} \sim_s \{ax_1 \rhd a, ax_2 \rhd \mathsf{enc}(c, b)\}$ since, intuitively, there is no way to open the ciphertexts or to construct them, hence no information on the content may leak.

2.5 Symbolic equivalence

Now we wish to check static equivalence on any possible trace. This is captured by the following definition:

Definition 5. *Let C and C' be two constraints whose corresponding frames are ϕ and ϕ'. C is* symbolically equivalent *to C', $C \approx_s C'$, if:*
- for all $(\theta, \sigma) \in \mathsf{Sol}(C)$, there exists σ' such that $(\theta, \sigma') \in \mathsf{Sol}(C')$, and $\phi\sigma \sim_s \phi'\sigma'$,
- for all $(\theta, \sigma') \in \mathsf{Sol}(C')$, there exists σ such that $(\theta, \sigma) \in \mathsf{Sol}(C)$, and $\phi\sigma \sim_s \phi'\sigma'$.

Example 8. As explained for instance in [3], the security of the handshake protocol against offline guessing attacks can be modeled as an equivalence property between two samples of the protocol instance, one in which, at the end of the protocol, the key is revealed and the other in which a random number is revealed instead. This amounts to check the symbolic equivalence of the two constraints:

– $C_1 = (\phi \cup \{ax_5, 5 \rhd k_{ab}\}, D \cup \{X_3, 5 \overset{?}{\vdash} y\}, \emptyset, \emptyset)$, and
– $C_2 = (\phi \cup \{ax_5, 5 \rhd k\}, D \cup \{X_3, 5 \overset{?}{\vdash} y\}, \emptyset, \emptyset)$

where D is as in Example 4 and ϕ is as in Example 3.

The constraints C_1 and C_2 are *not* symbolically equivalent: considering the assignment $\sigma = \{x \mapsto n_a, y \mapsto n_a\}$, there is a recipe $X_3\theta = \mathsf{dec}(ax_3, ax_5)$ yielding this solution, while any solution σ' of C_2 maps x to n_a and, if $X_3\theta = \mathsf{dec}(ax_3, ax_5)$, we must have $y\sigma'\!\downarrow = \mathsf{dec}(\mathsf{enc}(n_a, k_{ab}), k)$, which is not possible since this is not a constructor term.

Any trace equivalence problem can be expressed as an instance of the equivalence of an *initial pair of constraints*, that is a pair of the form (ϕ_1, D_1, E_1, E'_1), (ϕ_2, D_2, E_2, E'_2) in which:

– $E'_1 = E'_2 = \emptyset$, and E_1, E_2 only contain equations;
– $\phi_1 = \{ax_1, 1 \rhd u_1, \ldots, ax_m, m \rhd u_m\}$, and $D_1 = X_1, i_1 \overset{?}{\vdash} s_1;\ \ldots;\ X_n, i_n \overset{?}{\vdash} s_n$;
– $\phi_2 = \{ax_1, 1 \rhd v_1, \ldots, ax_m.m \rhd v_m\}$, and $D_2 = X_1, i_1 \overset{?}{\vdash} t_1;\ \ldots;\ X_n, i_n \overset{?}{\vdash} t_n$.

Or else it is a pair as above, in which one of the components is replaced with $\perp$.

In particular, the number of components in the frame and in the deducibility part are respectively identical in the two constraints, when none of them is $\perp$. *This will be an invariant in all our transformation rules.* Hence we will always assume this without further mention. This is unchanged by the transformations, unless the constraint becomes $\perp$. We keep the notation m for the size of the frames. Finally, the consistency of the frame after instanciation (the first condition of Definition 3) is satisfied for all solutions of initial constraints and is again an invariant, hence we will not care of this condition.

As explained in [14], such initial constraints are sufficient for our applications. The case where one of the component is $\perp$ solves the satisfiability problem for the constraint: the constraint solving procedure of [12] solves this specific instance.

3 Transformation rules

The main result of this paper is a decision procedure for symbolic equivalence of an initial pair of constraints:

Theorem 1. *Given an initial pair (C, C'), it is decidable whether $C \approx_s C'$.*

This result in itself is already known (e.g. [3, 9]), but, as claimed in the introduction, the known algorithms cannot yield any reasonable implementation. We propose here a new algorithm/proof, which is implemented. As pointed in [14], this yields a decision algorithm for the observational equivalence of simple processes without replication nor else branch. The class of simple processes captures most existing protocols.

The decision algorithm works by rewriting pairs of constraints, until a trivial failure or a trivial success is found. These rules are branching: they rewrite a pair of constraints into two pairs of constraints. Transforming the pairs of constraints therefore builds a binary tree. Termination requires to keep track of some information, that is recorded using flags, which we describe first. In Section 4, we show that the tree is then finite: the rules are terminating. The transformation rules are also correct: if all leaves are success leaves, then the original pair of constraints is equivalent. They are finally complete: if the two original constraints are equivalent then any of two pairs of constraints resulting from a rewriting steps are also equivalent.

3.1 Flags

The flags are additional constraints that restrict the recipes. We list them here, together with (a sketch of) their semantics.

Constraints $X, i \overset{?}{\vdash}_F u$ may be indexed with a set F consisting of propositions $\texttt{NoCons}_f$ where f is a constructor. Any solution (θ, σ) such that $X\theta$ is headed with f is then excluded. Expressions $\zeta, j \rhd_F u$ in a frame are indexed with a set F consisting of:

- $\texttt{NoCons}_f$ (as above) discards the solutions (θ, σ) such that a subterm of a recipe allows to deduce $u\sigma$ using f as a last step.
- $\texttt{NoDest}_f(i)$ where f is a destructor and $i \leq m$ discards the solutions (θ, σ) such that there exists $X, j \overset{?}{\vdash} v$ with $j \leq i$ and $\zeta'_2, \ldots, \zeta'_n$ where $f(\zeta\theta, \zeta'_2, \ldots, \zeta'_n)$ occurs as a subterm in $X\theta$, unless we use a shortcut explicitly given in the frame.
- $\texttt{NoUse}$. The corresponding elements of the frame cannot be used in any recipe, and avoids shifting the indices.

3.2 The rules

The rules are displayed in Figure 1 for single constraints. We explain in Section 3.3 how they are applied to pairs of constraints (an essential feature of our algorithm). A simple idea would be to guess the top function symbol of a recipe and replace the recipe variable with the corresponding instance. When the head symbol of a recipe is a constructor and the corresponding term is not a variable, this is nice, since the constraint becomes simpler. This is the purpose of the rule CONS. When the top symbol of a recipe is a destructor, the constraint becomes more complex, introducing new terms, which yields non-termination.

Our strategy is different for destructors: we switch roughly from the top position of the recipe to the *redex position*. Typically, in case of symmetric encryption, if a ciphertext is in the frame, we will guess whether the decryption key is deducible, and at which stage.

The CONS rule simply guesses whether the top symbol of the recipe is a constructor f. Either it is, and then we can split the constraint, or it is not and we add a flag forbidding this. The rule AXIOM also guesses whether a trivial recipe can be applied. If so, the constraint can simply be removed. Otherwise, it means that the right-hand-side of the deducibility constraint is different from the members of the frame. The DEST rule is more tricky. If v is a non-variable member of the frame, that can be unified with a non variable subterm of a left side of a rewrite rule (for instance v is a ciphertext), we guess whether the rule can be applied to v. This corresponds to the equation $u_1 \overset{?}{=} v$, that yields an instance of w, the right member of the rewrite rule, provided that the rest of the left member is also deducible: we get constraints $X_2, i \overset{?}{\vdash} u_2; \ldots; X_n, i \overset{?}{\vdash} u_n$. The flag `NoDest` is added in any case to the frame, since we either already applied the destructor, and this application result is now recorded in the frame by $f(\zeta, X_2, \ldots, X_n), i \rhd w$, or else it is assumed that f applied to v will not yield a redex.

The remaining rules cover the comparisons that an attacker could perform at various stages. The equality rules guess equalities between right sides of deducibility constraints and/or members of the frame. If a member of the frame is deducible at an early stage, then this message does not bring any new information to the attacker: it becomes useless, hence the `NoUse` flag.

Finally, the last rule is the only rule that is needed to get in addition a static equivalence decision algorithm, as in [1]. Thanks to this rule, if a subterm of the frame is deducible, then there will be a branch in which it is deduced.

3.3 How to use the transformation rules

In the previous section we gave rules that apply on a single constraint. We explain here how they are extended to pairs of constraints. If one of the constraint is $\perp$, then we proceed as if there was a single constraint. Otherwise, the indices i (resp. i_1, i_2) and the recipes X, ζ (resp. $X_1, X_2, \zeta_1, \zeta_2$) matching the left side of the rules *must be identical in both constraints*: we apply the rules at the same positions in both constraints.

We have to explain now what happens when, on a given pair (C, C') a rule can be applied on C and not on C' (or the converse).

$$\begin{tikzcd} & X_1, i \overset{?}{\vdash}_F t_1; \cdots ; X_n, i \overset{?}{\vdash}_F t_n; X \overset{?}{=} f(X_1,\ldots,X_n) \\ \underline{\textsc{Cons}} : X, i \overset{?}{\vdash}_F f(t_1,\ldots,t_n) \arrow[ur] \arrow[dr] & \\ & X, i \overset{?}{\vdash}_{F+\mathtt{NoCons}_f} f(t_1,\ldots,t_n) \end{tikzcd}$$

If $\mathtt{NoCons}_f \notin F$ and $X_1, \ldots X_n$ are fresh variables.

$$\begin{tikzcd} & u \overset{?}{=} v;\ X \overset{?}{=} \zeta \\ \underline{\textsc{Axiom}} : X, i \overset{?}{\vdash}_F v \arrow[ur] \arrow[dr] & \\ & X, i \overset{?}{\vdash}_F v;\ X \overset{?}{\neq} \zeta \end{tikzcd}$$

If $v \notin \mathcal{X}$, ϕ contains $\zeta, j \rhd_G u$ with $\mathtt{NoUse} \notin G$, and $i \geq j$.

$$\begin{tikzcd} & X_2, i \overset{?}{\vdash} u_2;\ \cdots ;\ X_n, i \overset{?}{\vdash} u_n;\ u_1 \overset{?}{=} v;\ \zeta, j \rhd_{G+\mathtt{NoDest}_f(m)} v; \\ \underline{\textsc{Dest}} : \zeta, y \rhd_G v \arrow[ur] \arrow[dr] & f(\zeta, X_2, \ldots, X_n), i \rhd w \\ & \zeta, j \rhd_{G+\mathtt{NoDest}_f(i)} v \end{tikzcd}$$

If $v \notin \mathcal{X}$, $\mathtt{NoUse} \notin G$, there is a rewrite rule $f(u_1, \ldots, u_n) \to w$, $k < i$ whenever $\mathtt{NoDest}_f(k) \in G$ and i is minimal such that $j \leq i$ and there is some constraint $X, i \overset{?}{\vdash} w$ ($i = m$ if there is no such constraint).

$$\begin{tikzcd} & \zeta_1, i_1 \rhd_{F_1} u_1;\ \zeta_2, i_2 \rhd_{F_2} u_1;\ u_1 \overset{?}{=} u_2 \\ \underline{\textsc{Eq-left-left}} : \zeta_1, i_1 \rhd_{F_1} u_1;\ \zeta_2, i_2 \rhd_{F_2} u_2 \arrow[ur] \arrow[dr] & \\ & \zeta_1, i_1 \rhd_{F_1} u_1;\ \zeta_2, i_2 \rhd_{F_2} u_2;\ u_1 \overset{?}{\neq} u_2 \end{tikzcd}$$

If $\mathtt{NoUse} \notin F_1 \cup F_2$ and $i_1 \leq i_2$.

$$\begin{tikzcd} & X_1 = X_2;\ u_1 \overset{?}{=} u_2 \\ \underline{\textsc{Eq-right-right}} : X_2, i_2 \overset{?}{\vdash} u_2 \arrow[ur] \arrow[dr] & \\ & X_2, i_2 \overset{?}{\vdash} u_2;\ u_1 \overset{?}{\neq} u_2 \end{tikzcd}$$

If $X_1, i_1 \overset{?}{\vdash} u_1;$ and $i_1 \leq i_2$.

$$\begin{tikzcd} & \zeta, j \rhd_{G+\mathtt{NoUse}} u;\ u \overset{?}{=} v \\ \underline{\textsc{Eq-left-right}} : \zeta, j \rhd_G v \arrow[ur] \arrow[dr] & \\ & \zeta, j \rhd_G v;\ u \overset{?}{\neq} v \end{tikzcd}$$

If $X, i \overset{?}{\vdash}_F u;$, $\mathtt{NoUse} \notin G$ and $j > i$.

$$\begin{tikzcd} & X_1, m \overset{?}{\vdash} u_1;\ \cdots ;\ X_n, m \overset{?}{\vdash} u_n; \\ \underline{\textsc{Ded-subterms}} : \zeta, i \rhd_F f(u_1, \ldots, u_n) \arrow[ur] \arrow[dr] & \zeta, i \rhd_{F+\mathtt{NoCons}_f} u \\ & \zeta, i \rhd_{F+\mathtt{NoCons}_f} f(u_1, \ldots, u_n) \end{tikzcd}$$

If $\mathtt{NoCons}_f, \mathtt{NoUse} \notin F$ and $X_1, \ldots, X_n$ are fresh variables.

All rules assume that the equations have a mgu and that this mgu is eagerly applied to the resulting constraint without yielding any trivial disequation.

Fig. 1. Transformation rules

Example 9. Let $C = (\phi, D, E, E')$ and $C' = (\phi, D', E, E')$ where $E = E' = \emptyset$, $\phi = ax_1, 1 \rhd a$, $D = X, 1 \overset{?}{\vdash} \mathsf{enc}(x_1, x_2)$, and $D' = X, 1 \overset{?}{\vdash} x$. The rule Cons can be applied on C and not on C'. However, we have to consider solutions where $\mathsf{enc}(x_1, x_2)\sigma$ and $x\sigma'$ are both obtained by a construction. Hence, it is important to enable this rule on both sides. For this, we first apply the substitution $x \mapsto \mathsf{enc}(y_1, y_2)$ where y_1, y_2 are fresh variables. This yields the two pairs of constraints (C_1, C_1') and (C_2, C_2') (forgetting about equations):

– $C_1 = (\phi, X_1, 1 \overset{?}{\vdash} x_1; X_2, 1 \overset{?}{\vdash} x_2)$ and $C_1' = (\phi, X_1, 1 \overset{?}{\vdash} y_1;\ X_2, 1 \overset{?}{\vdash} y_2)$;
– $C_2 = (\phi, X, 1 \overset{?}{\vdash}_{\mathtt{NoCons}_{\mathsf{enc}}} \mathsf{enc}(x_1, x_2))$ and $C_2' = (\phi, X, 1 \overset{?}{\vdash}_{\mathtt{NoCons}_{\mathsf{enc}}} x)$.

Therefore, the rule Cons, (this is similar for Ded-subterms), when applied to pairs of constraints comes in three versions: either the rule is applied on both sides or, if $X, i \overset{?}{\vdash} f(t_1, \ldots, t_n)$ (resp. $\zeta \rhd f(t_1, \ldots, t_n)$) is in C, and $X, i \overset{?}{\vdash} x$ (resp. $\zeta \rhd x$) is in C', we may apply the rule on the pair of constraints, adding to C' the equation $x \overset{?}{=} f(x_1, \ldots, x_n)$ where $x_1, \ldots, x_n$ are fresh variables. The third version is obtained by switching C and C'. This may introduce new variables, that yield a termination issue, which we discuss in Section 4.1. Similarly, the rules Axiom and Dest assume that $v \notin \mathcal{X}$. This has to be satisfied by C or C'. In case of the rule Dest, this means that the variables of the rewrite rule might not be immediately eliminated: this may also introduce new variables. For the rules Eq-left-left, Eq-right-right and Eq-left-right, we require that at least one new non-trivial equality (or disequality) is added to one of the two constraints (otherwise there is a trivial loop).

For all rules, if a rule is applicable on one constraint and not the other, we do perform the transformation, however replacing a constraint with $\perp$ when a condition becomes false or meaningless. Furthermore, we also replace a constraint C with $\perp$ when:

– the rule Dest cannot be applied on C; and
– C contains a constraint $X, i \overset{?}{\vdash} v$ such that v is not a variable and the rules Cons and Axiom cannot be applied to it.

Altogether this yields a transformation relation $(C, C') \to (C_1, C_1'), (C_2, C_2')$ on pairs of constraints: a node labeled (C, C') has two sons, respectively labeled (C_1, C_1') and (C_2, C_2').

Our algorithm can be stated as follows:

– Construct, from an initial pair of constraints (C_0, C_0') a tree, by applying as long as possible a transformation rule to a leaf of the tree.
– If, at some point, there is a leaf to which no rule is applicable and that is labeled $(C, \perp)$ or $(\perp, C)$ where $C \neq \perp$, then we stop with $C_0 \not\approx_s C_0'$.
– Otherwise, if the construction of the tree stops without reaching such a failure, return $C_0 \approx_s C_0'$.

Our algorithm can also be used to decide static equivalence of frames, as well as the (un)satisfiability of a constraint. Furthermore, in case of failure, a witness of the failure can be returned, using the equations of the non-$\perp$ constraint.

4 Correctness, completeness and termination

4.1 Termination

In general, the rules might not terminate, as shown by the following example:

Example 10. Consider the initial pair of contraints (C, C') given below:

$$C = \begin{cases} a \overset{?}{\vdash} \mathsf{enc}(x_1, x_2) \\ a, b \overset{?}{\vdash} x_1 \end{cases} \qquad C' = \begin{cases} a \overset{?}{\vdash} y_1 \\ a, b \overset{?}{\vdash} \mathsf{enc}(y_1, y_2) \end{cases}$$

We may indeed apply Cons yielding (on one branch):

$$C_1 = \begin{cases} a \overset{?}{\vdash} x_1 \\ a \overset{?}{\vdash} x_2 \\ a, b \overset{?}{\vdash} x_1 \end{cases} \qquad C_1' = \begin{cases} a \overset{?}{\vdash} z_1 \\ a \overset{?}{\vdash} z_2 \qquad \text{and } y_1 \overset{?}{=} \mathsf{enc}(z_1, z_2) \\ a, b \overset{?}{\vdash} \mathsf{enc}(\mathsf{enc}(z_1, z_2), y_2) \end{cases}$$

Then, again using Cons, we get back as a subproblem the original constraints.

Fortunately, there is a simple complete strategy that avoids this behavior, by breaking the symmetry between the two constraints components. We assume in the following that, applying

- Cons to (C, C') where $X, i \overset{?}{\vdash} x \in C$ and $X, i \overset{?}{\vdash} f(t_1, \ldots, t_n) \in C'$,
- Ded-subterms to (C, C') where $\zeta, j \rhd x \in C$ and $\zeta, j \rhd f(t_1, \ldots, t_n) \in C'$,
- Dest to (C, C') where $X, i \overset{?}{\vdash} u; \zeta, j \rhd x \in C$ and $X, i \overset{?}{\vdash} u'; \zeta, j \rhd v' \in C'$

are only allowed when no other rule can be applied.

There is however no such restriction, when we switch the elements of the pair. If we come back to Example 10, we still apply the same transformation rule to the pair (C, C'), but we cannot apply Cons to (C_1, C_1') since Eq-right-right can be applied to the constraint C_1, yielding a failure: $C \not\approx_s C'$.

Lemma 1. *With the above strategy, the transformation rules are terminating on any initial pair of constraint systems.*

Idea of the proof: as long as no new first-order variable is introduced, the set of first-order terms appearing in the constraint is roughly bounded by the subterms of the constraint. (This relies on the properties of the rewrite system). Loops are then prevented by the flags. Now, because of the eager application of substitutions, the only cases in which new first-order variables are introduced are the above cases of applications of Cons, Ded-subterms and Dest. Until new variables are introduced in the right constraints, the above argument applies:

the sequence of transformations is finite. Then, according to the strategy, when new variables are introduced on the right constraint, no other rule may apply. This implies that the left constraint (considered in isolation) is irreducible: it is of the form $X_1, i_1 \overset{?}{\vdash} x_1, \ldots, X_n, i_n \overset{?}{\vdash} x_n, \ldots$ where $x_1, \ldots x_n$ are distinct variables (which we call a *solved constraint*). From this point onwards, the rules Dest, Ded-subterms will never be applicable and therefore, no element will be added to the frames. Then, either usable elements of the frames are strictly decreasing (using a Eq-left-right) or else we preserve the property of being solved on the left. In the latter case, the first termination argument can be applied to the right constraint.

4.2 Correctness

The transformation rules yield a finite tree labeled with pairs of constraints.

Lemma 2. *If all leaves of a tree, whose root is labeled with (C_0, C_0') (a pair of initial constraints), are labeled either with $(\bot, \bot)$ or with some (C, C') with $C \neq \bot, C' \neq \bot$, then $C_0 \approx_s C_0'$.*

The idea of the proof is to first analyse the structure of the leaves. We introduce a restricted symbolic equivalence $\approx_s^r$ such that $C \approx_s^r C'$ for any leaf whose two label components are distinct from $\bot$. Roughly, this restricted equivalence will only consider the recipes that satisfied the additional constraints induced by the flags. Then we show that $\approx_s^r$ is preserved upwards in the tree: for any transformation rule, if the two pairs of constraints labeling the sons of a node are respectively in $\approx_s^r$, then the same property holds for the father. Finally, $\approx_s^r$ coincides with $\approx_s$ on the initial constraints (that contain no flag).

4.3 Completeness

We prove that the symbolic equivalence is preserved by the transformation rules, which yields:

Lemma 3. *If (C_0, C_0') is a pair of initial constraints such that $C_0 \approx_s C_0'$, then all leaves of a tree, whose root is labeled with (C_0, C_0'), are labeled either with $(\bot, \bot)$ or with some (C, C') with $C \neq \bot$ and $C' \neq \bot$.*

5 Implementation and experiments

An Ocaml implementation of an early version of the procedure described in this paper, as well as several examples, are available at `http://www.lsv.ens-cachan.fr/~cheval/programs/index.php` (around 5000 lines of Ocaml). Our implementation closely follows the transformation rules that we described. For efficiency reasons, a strategy on checking the rules applicability has been designed in addition.

We checked the implementation on examples of static equivalence problems, on examples of satisfiability problems, and on symbolic equivalence problems that come from actual protocols. On all examples the tool terminates in less than a second (on a standard laptop). Note that the input of the algorithm is a pair of constraints: checking the equivalence of protocols would require in addition an interleaving step, that could be expensive.

We have run our tool on the following family of examples presented in [5]:

$$\phi_n = \{ax_1 \triangleright t_n^0, ax_2 \triangleright c_0, ax_3 \triangleright c_1\} \text{ and } \phi'_n = \{ax_1 \triangleright t_n^1, ax_2 \triangleright c_0, ax_3 \triangleright c_1\}$$

where $t_0^i = c_i$ and $t_{n+1}^i = \langle \mathsf{enc}(t_n^i, k_n^i), k_n^i \rangle$, $i \in \{0, 1\}$. In these examples, the size of the distinguishing tests increase exponentially while the sizes of the frames grow linearly. As KiSs [10], our tool outperforms YAPA [4] on such examples.

For symbolic equivalences, we cannot compare with other tools (there is no such tools); we simply tested the program on some home made benchmarks as well as on the handshake protocol, several versions of the encrypted password transmission protocol, the encrypted key exchange protocol [13], each for the offline guessing attack property. We checked also the strong secrecy for the corrected Dennin-Sacco key distribution protocol. Unfortunately we cannot (yet) check anonymity properties for e-voting protocols, as we would need to consider more cryptographic primitives.

6 Conclusion

We presented a new algorithm for deciding symbolic equivalence, which performs well in practice. There is still some work to do for extending the results and the tool. First, we use trace equivalence, which requires to consider all interleavings of actions; for each such interleaving, a pair of constraints is generated, which is given to our algorithm. This requires an expensive overhead (which is not implemented), that might be unnecessary. Instead, we wish to extend our algorithm, considering pairs of sets of constraints and use a symbolic bisimulation. This looks feasible and would avoid the detour through trace equivalence. This would also allow drop the determinacy assumption on the protocols and to compare our method with ProVerif [7].

We considered only positive protocols; we wish to extend the algorithm to non-positive protocols, allowing disequality constraints from the start. Finally, we need to extend the method to other cryptographic primitives, typically blind signatures and zero-knowledge proofs.

Acknowledgments. We wish to thank Sergiu Bursuc for fruitful discussions.

References

1. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 367(1–2):2–32, 2006.

2. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. of 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, 2001.
3. M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proc. of 12th ACM Conference on Computer and Communications Security*, 2005.
4. M. Baudet. YAPA (Yet Another Protocol Analyzer), 2008. `http://www.lsv.ens-cachan.fr/~baudet/yapa/index.html`.
5. M. Baudet, V. Cortier, and S. Delaune. YAPA: A generic tool for computing intruder knowledge. In *Proc. of 20th International Conference on Rewriting Techniques and Application (RTA'09)*, LNCS, 2009.
6. B. Blanchet. An automatic security protocol verifier based on resolution theorem proving (invited tutorial). In *Proc. of 20th International Conference on Automated Deduction (CADE'05)*, 2005.
7. 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.
8. V. Cheval, H. Comon-Lundh, and S. Delaune. Automating security analysis: symbolic equivalence of constraint systems. Technical report, `http://www.lsv.ens-cachan.fr/~cheval/programs/technical-report.pdf`, 2010.
9. Y. Chevalier and M. Rusinowitch. Decidability of symbolic equivalence of derivations. Unpublished draft, 2009.
10. Ş. Ciobâcă. KiSs, 2009. `http://www.lsv.ens-cachan.fr/~ciobaca/kiss`.
11. H. Comon-Lundh. Challenges in the automated verification of security protocols. In *Proc. of 4th International Joint Conference on Automated Reasoning (IJCAR'08)*, volume 5195 of *LNAI*, pages 396–409, Sydney, Australia, 2008. Springer-Verlag.
12. H. Comon-Lundh, V. Cortier, and E. Zalinescu. Deciding security properties of cryptographic protocols. application to key cycles. *Transaction on Computational Logic*, 11(2), 2010.
13. R. Corin, J. Doumen, and S. Etalle. Analysing password protocol security against off-line dictionary attacks. *Electr. Notes Theor. Comput. Sci.*, 121:47–63, 2005.
14. V. Cortier and S. Delaune. A method for proving observational equivalence. In *Proc. of 22nd Computer Security Foundations Symposium (CSF'09)*, pages 266–276. IEEE Comp. Soc. Press, 2009.
15. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.
16. U. Fendrup, H. Hüttel, and J. N. Jensen. Modal logics for cryptographic processes. *Theoretical Computer Science*, 68, 2002.
17. H. Huttel. Deciding framed bisimulation. In *4th International Workshop on Verification of Infinite State Systems INFINITY'02*, pages 1–20, 2002.
18. C. Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.
19. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. of 8th ACM Conference on Computer and Communications Security*, 2001.
20. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is np-complete. In *Proc. of 14th Computer Security Foundations Workshop*, 2001.
21. C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *Proc. of 16th Conference on Automated Deduction*, volume 1632, pages 314–328. LNCS, 1999.

Automated verification of equivalence properties of cryptographic protocols[*]

Rohit Chadha[1], Ştefan Ciobâcă[1], and Steve Kremer[1,2]

[1] LSV, ENS Cachan & CNRS & INRIA
[2] INRIA Nancy - Grand-Est

Abstract. Indistinguishability properties are essential in formal verification of cryptographic protocols. They are needed to model anonymity properties, strong versions of confidentiality and resistance to offline guessing attacks, and can be conveniently modeled using process equivalences. We present a novel procedure to verify equivalence properties for bounded number of sessions. Our procedure is able to verify trace equivalence for determinate cryptographic protocols. On determinate protocols, trace equivalence coincides with observational equivalence which can therefore be automatically verified for such processes. When protocols are not determinate our procedure can be used for both under- and over-approximations of trace equivalence, which proved successful on examples. The procedure can handle a large set of cryptographic primitives, namely those which can be modeled by an optimally reducing convergent rewrite system. Although, we were unable to prove its termination, it has been implemented in a prototype tool and has been effectively tested on examples, some of which were outside the scope of existing tools.

1 Introduction

Cryptographic protocols are distributed programs which rely on the use of cryptography to secure electronic transactions such as those that arise in electronic commerce and wireless communication. They are also being applied in new domains such as in Internet voting—legally binding political elections in Estonia, Norway and Switzerland offer the possibility for Internet voting in 2011. This has led to increasing demands on the complexity of desired security properties, leading to more complex cryptographic protocols. Given the socio-economic-political consequences and the history of incorrect design of cryptographic protocols, the need for formal proofs of correctness of protocols has been widely recognized. Formal reasoning about cryptographic protocols is challenging as one has to reason against all potentially malicious behavior—all communication between protocol participants is assumed to be under the control of an adversary.

In order to make the task of formal analysis amenable to automation, usually the assumption of black-box cryptography and unbounded computational power

[*] This work was partially supported by ANR projects ARA SESUR AVOTÉ and JCJC VIP no 11 JS02 006 01 and the ERC grant agreement n^o 258865, project ProSecure.

of the adversary is made. This adversarial model is often called the Dolev-Yao model and is derived from Dolev and Yao's seminal paper [29]. It has proved extremely successful, and there are several automated tools [10, 6, 31] that can automatically check trace-properties such as (weak forms of) confidentiality and authentication. While trace-based properties are certainly important, many crucial security properties can only be expressed in terms of *indistinguishability* (or equivalence). They include strong flavors of confidentiality [11]; resistance to guessing attacks in password based protocols [8]; and anonymity properties in private authentication [3], electronic voting [26, 7], vehicular networks [24] and RFID protools [5, 15]. More generally, indistinguishability allows to model security by the means of ideal systems, which are correct by construction [4, 25]. Indistinguishability properties of cryptographic protocols are naturally modeled by the means of *observational* and *testing equivalences* in cryptographic extensions of process calculi, e.g., the spi [4] and the applied-pi calculus [2]. While we have good tools for automated verification of trace properties, the situation is different for indistinguishability properties.

State-of-the-art. Hüttel [34] showed undecidability of observational equivalence in the spi calculus, even for the finite control fragment, as well as decidability for the finite, i.e., replication-free, fragment of the spi calculus. The decidability result however only holds for a fixed set of cryptographic primitives and does not yield a practical algorithm. Current results [12] allow to approximate observational equivalence for an unbounded number of sessions. However, this approximation does not suffice to conclude for many applications, *e.g.*, [26, 5]. Our approach overcomes these limitations for some applications in [26]. We still cannot conclude for the e-passport example in [5], albeit for a different reason: our procedure does not currently handle else branches in protocols.

Symbolic bisimulations have also been devised for the spi [14, 13, 39] and applied pi calculus [27, 35] to avoid unbounded branching due to adversary inputs. However, only [27, 39] and [14] yield a decision procedure, again only approximating observational equivalence. The results of [27] have been further refined to show a decision procedure on a restricted class of *simple* processes [23]. They rely on a procedure deciding the equivalence of constraint systems, introduced by Baudet [8], for the special case of verifying the existence of guessing attacks. Baudet's procedure allows arbitrary cryptographic primitives that can be modeled as a subterm convergent rewrite systems [1]. An alternate procedure achieving the same goal was proposed by Chevalier and Rusinowitch [19]. However, both procedures are highly non-deterministic and do not yield a reasonable algorithm that could be implemented. Therefore, Cheval *et al.* [17] have designed a new procedure and a prototype tool to decide the equivalence of constraint systems, but only for a fixed set of primitives. Tools have also been implemented for checking testing equivalence [30], open bisimulation [39] and trace equivalence [18] for a bounded number of sessions but again only for a limited set of primitives. One may note that [18] is the only decision procedure to consider negative tests (else branches), crucial in several case studies [5, 3].

Our contribution. We introduce a new procedure for verifying equivalence properties for processes specified in a cryptographic process calculus (without replication). Our main contributions are as follows.

– Our procedure checks for two equivalences which over- and under-approximate the standard notion of trace equivalence $\approx_t$ for cryptographic protocols: the under-approximation can be used to prove protocols correct while the over-approximation can be used to rule out incorrect protocols.
– Cortier and Delaune [23] have shown that observational equivalence coincides with $\approx_t$ for the class of *determinate* processes. They also give a decision procedure for a strict sub-class of determinate processes, namely, *simple* processes. We show that for determinate processes the coarser relation coincides with $\approx_t$, and our procedure can be used to verify observational equivalence for the whole class of determinate processes.
– A novelty of our procedure is that it is based on a *fully abstract* modeling of symbolic traces in *first-order Horn clauses*. This is in contrast to the constraint-solving techniques employed in [39, 17, 18, 8, 19] for verifying under-approximations of observational equivalence. Techniques based on Horn clauses have been extensively used, e.g., in [10, 40, 33], for an unbounded number of sessions. Of these tools, only ProVerif [10, 12] can verify an equivalence property, which is an under-approximation of observational equivalence. Horn clause modeling of an unbounded number of sessions of security protocols may allow false attacks. In contrast, we show our modeling of a bounded number of sessions for determinate protocols to be precise.
– Our modeling is fully abstract for arbitrary cryptographic primitives that can be modeled as a convergent rewrite system which has the *finite variant property*. Not only this strictly includes the class of primitives that can be modeled as subterm convergent rewrite systems, but this also allows us to handle a larger class of cryptographic primitives than [39, 17, 18, 8, 19, 10]. For example, this allows us to handle trapdoor commitment as used by Okamoto for electronic voting in [38]. Although we were unable to prove termination of our procedure, we conjecture it to terminate for the class of cryptographic primitives that can be modeled as subterm convergent rewrite systems. Our conjecture is supported by experimental evidence.
– Our procedure is implemented in the AKiSs (Active Knowledge in Security protocols) prototype tool and used among others to give the first automated proof of anonymity for the electronic voting protocol presented in [32].

Technical proofs are given in an accompanying technical report [16].

2 Preliminaries

Terms. Let $\mathcal{F}$ be a signature, i.e., a finite set of function symbols and *ar* a function that assigns to each function symbol a natural number, its arity. A function symbol of arity 0 is called a *constant.* Given a set of *atoms* $\mathcal{A}$ and a signature $\mathcal{F}$, we denote by $\mathcal{T}_{\mathcal{F},\mathcal{A}}$ the set of terms built inductively from $\mathcal{A}$ by applying functions symbols in $\mathcal{F}$. Given sets of atoms $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$, we denote the set $\mathcal{T}_{\mathcal{F}, \cup_{1 \leq i \leq n} \mathcal{A}_i}$ by $\mathcal{T}_{\mathcal{F}, \mathcal{A}_1, \mathcal{A}_2, \ldots \mathcal{A}_n}$. We assume that we have the following countably infinite pairwise disjoint sets: a set $\mathcal{N}$ of *private names*, $\mathcal{M}$ of *public names*, a set $\mathcal{C}$ of *public channel names*, a set $\mathcal{W}$ of *parameters*, and a set $\mathcal{X}$ of *message variables*. Intuitively, elements of the set $\mathcal{N}$ represent nonces generated by honest principals of a protocol, elements of $\mathcal{M}$ represent nonces available both to the adversary and to the honest participants and elements of $\mathcal{C}$ represent names of public channels (e.g. the name of a public network). Elements of $\mathcal{W}$ are pointers used by the adversary to refer to messages output by the honest participants in a protocol. We fix an enumeration $w_1, w_2, \ldots$ of the elements of $\mathcal{W}$. We let x, y, z range over $\mathcal{X}$. We also define the following set of terms:

- Terms denotes the set of all terms $\mathcal{T}_{\mathcal{F}, \mathcal{N}, \mathcal{M}, \mathcal{W}, \mathcal{X}}$.
- Messages denotes the set of *messages* $\mathcal{T}_{\mathcal{F}, \mathcal{N}, \mathcal{M}}$.
- SMessages denotes the set of *symbolic messages* $\mathcal{T}_{\mathcal{F}, \mathcal{N}, \mathcal{M}, \mathcal{X}}$.

If t is a term, we denote by $vars(t)$ the set of variables appearing in t, by $names(t)$ the set of names (public or private) appearing in t. The functions $vars$, $names$ are extended to sequences and sets of terms as expected.

Example 1. Consider the signature $\mathcal{F} = \{\mathsf{enc}, \mathsf{dec}, \mathsf{pair}, \mathsf{fst}, \mathsf{snd}\}$. The term $t = \mathsf{pair}(\mathsf{enc}(a, k_1, r_1), \mathsf{enc}(b, k_2, r_2))$ models the pair of the asymmetric encryptions of public names a and b with keys k_1, resp. k_2 and randomness r_1, resp. r_2.

A substitution is a partial function $\sigma : \mathcal{W} \cup \mathcal{X} \to \mathsf{Terms}$. We restrict substitutions to map elements of $\mathcal{W}$ to elements of Messages and elements of $\mathcal{X}$ to elements of SMessages. The domain of σ shall be denoted by $dom(\sigma)$. We denote by $\sigma[X]$ the substitution whose domain is restricted to X. We only consider substitutions with finite domains. As usual, a substitution extends homomorphically to terms and we write $t\sigma$ for the term obtained by applying σ to t.

Rewriting and unification. Two terms s and t are (syntactically) *unifiable* if there exists a substitution σ such that $s\sigma = t\sigma$. We denote by $\mathrm{mgu}(s, t)$ their most general unifier. We assume that the reader is familiar with basic notions of rewriting and only briefly introduce our notations. A rewrite system R is a set of rewrite rules of the form $\ell \to r$ where $\ell, r \in \mathsf{Terms}$, $names(l, r) = \emptyset$ and $vars(r) \subseteq (\ell)$. We write $t \to_{\mathsf{R}} u$ when a term t can be rewritten in one step to u. $\to_{\mathsf{R}}^*$ denotes the transitive and reflexive closure of $\to_{\mathsf{R}}$. We only consider convergent rewrite systems and denote by $t\downarrow_{\mathsf{R}}$ the normal form of a term t. Two terms s and t are said to be equal modulo R, written $s =_{\mathsf{R}} t$, if $s\downarrow_{\mathsf{R}} = t\downarrow_{\mathsf{R}}$. Given a substitution σ, $\sigma\downarrow_R$ is the substitution such that $dom(\sigma\downarrow_{\mathsf{R}}) = dom(\sigma)$ and for all $u \in dom(\sigma)$, $\sigma\downarrow_R(u) = \sigma(u)\downarrow_{\mathsf{R}}$. We shall omit R when clear from the context.

Example 2. Let $\mathcal{F}$ be the signature in Example 1. Consider the rewrite system $\mathsf{R} = \{\mathsf{dec}(\mathsf{enc}(x, y, z), y) \to x, \mathsf{fst}(\mathsf{pair}(x, y)) \to x, \mathsf{snd}(\mathsf{pair}(x, y)) \to y\}$. The first rule models that a message can be decrypted, provided decryption uses the same key (represented by variable y) as encryption. The last two rules model projection of the first and second component of a pair. We have that $t = \mathsf{fst}(\mathsf{pair}(\mathsf{dec}(\mathsf{enc}(a, k, r), k), b)) \to_{\mathsf{R}} \mathsf{fst}(\mathsf{pair}(a, b)) \to_{\mathsf{R}} a = t\downarrow_{\mathsf{R}}$.

We recall the notion of complete set of variants for a convergent rewrite system [22]:

Definition 1. *A set of substitutions* variants$(t_1, \ldots, t_k)$ *is called a* complete set of variants *of terms* $t_1, \ldots, t_k$ *if for any substitution* ω *there exist* $\sigma \in$ variants$(t_1, \ldots, t_k)$ *and a substitution* τ *such that for all* $1 \leq j \leq k$ *we have that* $\omega[vars(t_j)]{\downarrow} = (\sigma{\downarrow}\tau)[vars(t_j)]$ *and* $(t_j\omega){\downarrow} = (t_j\sigma){\downarrow}\tau$.

Intuitively, the set of variants of t represents a pre-computation such that any instance of t in normal form is *syntactically* equal to an instance of $t\sigma_i{\downarrow}$ for some i, without the need to apply further rewrite steps. A rewrite system has the *finite variant property* if for any finite sequence of terms a finite, complete set of variants exists. An algorithm for computing complete sets of variants which is correct whenever the rewrite system is *optimally reducing* [37] is presented in [21]. Optimally reducing rewrite systems include subterm convergent systems [1] (and hence the classical Dolev Yao theories for encryption, signatures and hash functions), as well as a theory for modeling blind signatures. Complete sets of variants can be used to compute finite complete sets of unifiers modulo R [21], which are formally defined in [16] and denoted by $\mathrm{mgu}_{\mathsf{R}}$. We assume, henceforth, that rewrite systems in this paper have the finite variant property.

Frames, deducibility and static equivalence. We will use the notion of a *frame* [2] to represent messages which have been recorded by an attacker.

Definition 2. *A* frame φ *is a substitution* $\{w_1 \mapsto t_1, \ldots, w_n \mapsto t_n\}$ *where* $t_i \in$ Messages *(*$1 \leq i \leq n$*).*

Please note, in our definition, every frame φ with $|dom(\varphi)| = n$ has $dom(\varphi) = \{w_1, \ldots, w_n\}$. The set of all frames is denoted as Frames. The adversary can use the messages learnt from the run of a protocol to construct new messages. This is modeled as the deducibility relation.

Definition 3. *Any term in* $\mathcal{T}_{\mathcal{F},\mathcal{M},\mathcal{W}}$ *is said to be a recipe. We say that* a message t is deducible from φ with a recipe r *(written as* $\varphi \vdash^r t$*) if* $t \in$ Messages *and* $r\varphi =_{\mathsf{R}} t$*. We write* Recipes *for the set* $\mathcal{T}_{\mathcal{F},\mathcal{M},\mathcal{W}}$.

Example 3. Consider the signature $\mathcal{F}$ and the rewrite system R in Example 2. Let $\varphi = \{w_1 \mapsto \mathsf{enc}(s,k,r), w_2 \mapsto k\}$ where $s, k, r \in \mathcal{N}$ are private names. We have that $\varphi \vdash^{\mathsf{dec}(w_1,w_2)} s$. Note that $\mathsf{dec}(w_1, k) \notin$ Recipes as $k \in \mathcal{N}$. If s were public instead of being private (ie, $s \in \mathcal{M}$ instead of $s \in \mathcal{N}$) then we also have that $\varphi \vdash^s s$; as public names are always deducible.

Static equivalence captures indistinguishability of sequences of messages:

Definition 4. *Let* $r_1, r_2 \in$ Recipes*. A* test $r_1 \overset{?}{=} r_2$ holds *in a frame* φ *(written* $(r_1 = r_2)\varphi$*) if* $\varphi \vdash^{r_1} t$ *and* $\varphi \vdash^{r_2} t$ *for some* t, i.e., r_1 *and* r_2 *are recipes for the same term in* φ.

A frame φ_1 *is* statically included *in* φ_2 *(written* $\varphi_1 \sqsubseteq_s \varphi_2$*) iff for all* $r_1, r_2 \in$ Recipes *we have that* $(r_1 = r_2)\varphi_1$ *implies* $(r_1 = r_2)\varphi_2$*. Two frames* φ_1 *and* φ_2 *are* statically equivalent *(written* $\varphi_1 \approx_s \varphi_2$*) iff* $\varphi_1 \sqsubseteq_s \varphi_2$ *and* $\varphi_2 \sqsubseteq_s \varphi_1$.

Example 4. Let $a, b \in \mathcal{M}$ and $r, k, k' \in \mathcal{N}$. We have that $\{w_1 \mapsto \mathsf{enc}(a, k, r), w_2 \mapsto k\} \not\approx_s \{w_1 \mapsto \mathsf{enc}(b, k, r), w_2 \mapsto k\}$ because the test $(dec(w_1, w_2) = a)$ distinguishes the two frames. However, $\{w_1 \mapsto \mathsf{enc}(a, k, r), w_2 \mapsto k'\} \approx_s \{w_1 \mapsto \mathsf{enc}(b, k, r), w_2 \mapsto k'\}$. Moreover, we have that $\{w_1 \mapsto a, w_2 \mapsto b\} \sqsubseteq_s \{w_1 \mapsto a, w_2 \mapsto a\}$ while $\{w_1 \mapsto a, w_2 \mapsto a\} \not\sqsubseteq_s \{w_1 \mapsto a, w_2 \mapsto b\}$.

3 A cryptographic process calculus

We model cryptographic protocols using a simple process calculus which has similarities with the applied pi-calculus [2].

Syntax. We model a bounded number of instances of a cryptographic protocol as a *finite* set of traces. Traces are defined using sequences of *actions* generated by the following grammar:

$$a ::= \mathbf{in}(c, x) \;\mid\; \mathbf{out}(c, t) \;\mid\; [s \stackrel{?}{=} t]$$

where $x \in \mathcal{X}$, $s, t \in \mathsf{SMessages}$, $c \in \mathcal{C}$. A *trace* T is a sequence of actions $T = a_1.a_2.\dots.a_n$. As usual, a receive action $\mathbf{in}(c, x)$ acts as a binding construct for x. We assume the usual definitions of free and bound variables for traces. We also assume that each variable is bound at most once. A trace is *ground* if it does not contain any free variables. The set of ground traces shall be represented as $\mathsf{GndTraces}$. A set of traces $P = \{T_1, \dots, T_n\}$ is said to be a *process*. A process is ground if all of its traces are ground. We identify traces with singleton processes.

Remark 1. We do not have an ν operator: the binding happens implicitly by the use of private names in $\mathcal{N}$. We have also not explicitly included the parallel operator $|$ and the choice operator $+$. One could include these and generate the corresponding set of traces. Thus, there is no loss in expressivity. However, an explicit enumeration of the traces can result in an exponential number of traces.

Semantics. The semantics of a process is defined using the semantics of its traces. The semantics of a trace is given in terms of a labeled transition system T. We assume that all interactions between protocol participants are mediated by the adversary. The labeled transition system records the interaction of the protocol participants with the adversary. The set of labels of T is defined using the set $\mathsf{Recipes}$. Recall that the set $\mathsf{Recipes}$ is the set $\mathcal{T}_{\mathcal{F},\mathcal{M},\mathcal{W}}$ (see Section 2). The set of labels, Labels, is $\{\, \mathbf{in}(c, r), \mathbf{out}(c), \mathbf{test} \mid r \in \mathsf{Recipes}, c \in \mathcal{C}\,\}$.

The labeled transition system T is a subset of $(\mathsf{GndTraces} \times \mathsf{Frames}) \times \mathsf{Labels} \times (\mathsf{GndTraces} \times \mathsf{Frames})$. We write $(T, \varphi) \xrightarrow{\ell} (T', \varphi')$ whenever $((T, \varphi), \ell, (T', \varphi')) \in \mathsf{T}$. The frame in the transition system is used to record the messages that the protocol participants have sent in the past. The relation $\xrightarrow{\ell}$ is defined as follows:

$$\textsc{Receive}\ \frac{\varphi \vdash^r t}{(\mathbf{in}(c, x).T, \varphi) \xrightarrow{\mathbf{in}(c,r)} (T\{x \mapsto t\}, \varphi)} \qquad \textsc{Test}\ \frac{s =_{\mathsf{R}} t}{([s \stackrel{?}{=} t].T, \varphi) \xrightarrow{\mathbf{test}} (T, \varphi)}$$

$$\textsc{Send}\ \frac{}{(\mathbf{out}(c, t).T, \varphi) \xrightarrow{\mathbf{out}(c)} (T, \varphi \cup \{w_{|dom(\varphi)|+1} \mapsto t\})}$$

The label $\mathbf{in}(c, r)$ indicates a message sent by the adversary over the channel c and r is the recipe that adversary uses to create this message. The label $\mathbf{out}(c)$ indicates a message sent over the public channel c and transition rule SEND records the message sent in the frame. Finally, the rule Test is an internal action.

We write $(T, \varphi) \overset{\ell}{\Rightarrow} (T', \varphi')$ when either $(T, \varphi) \xrightarrow{\mathbf{test}^*, \ell, \mathbf{test}^*} (T', \varphi')$ and $\ell \neq \mathbf{test}$ or $(T, \varphi) \xrightarrow{\mathbf{test}^*} (T', \varphi')$ and $\ell = \mathbf{test}$, where $\mathbf{test}^*$ denotes an arbitrary number of $\mathbf{test}$ actions. We also write $(T_0, \varphi_0) \xrightarrow{\ell_1, \ldots, \ell_n} (T_n, \varphi_n)$ when $(T_0, \varphi_0) \xrightarrow{\ell_1} (T_1, \varphi_1) \ldots \xrightarrow{\ell_n} (T_n, \varphi_n)$ (and similarly for the $\Rightarrow$ relation) and say that $\ell_1 \ldots \ell_n$ is a *run* of (T_0, φ_0). If P is a process, we write $(P, \varphi) \xrightarrow{\ell_1, \ldots, \ell_n} (T', \varphi')$ (resp. $\xRightarrow{\ell_1, \ldots, \ell_n} (T', \varphi')$) if there exists a trace $T \in P$ such that $(T, \varphi) \xrightarrow{\ell_1, \ldots, \ell_n} (T', \varphi')$ (resp. $(T, \varphi) \xRightarrow{\ell_1, \ldots, \ell_n} (T', \varphi')$).

Process equivalences. We will now define different flavors of trace equivalence which will be useful in this paper. We first recall the standard definition of trace equivalence in cryptographic process algebras.

Definition 5. ***(Trace equivalence)*** *A ground process P is said to be* trace-included *in a ground process Q (written $P \sqsubseteq_t Q$) if whenever $(P, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (T, \varphi)$ then there exist T', φ' such that $(Q, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (T', \varphi')$ and $\varphi \approx_s \varphi'$. Two processes P and Q are* trace-equivalent *(written $P \approx_t Q$) if $P \sqsubseteq_t Q$ and $Q \sqsubseteq_t P$.*

We will also define two other notions of trace equivalence, one coarser and one more fine-grained.

Definition 6. *Given ground processes P and Q, we say that $P \sqsubseteq_{ct} Q$ if whenever $(P, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (T, \varphi)$ then there exist T', φ' such that $(Q, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (T', \varphi')$ and $\phi \sqsubseteq_s \phi'$. We say that $P \approx_{ct} Q$ if $P \sqsubseteq_{ct} Q$ and $Q \sqsubseteq_{ct} P$.*

The following example illustrates the difference between $\approx_t$ and $\approx_{ct}$.

Example 5. Let P and Q be the ground processes defined as follows: $P = \{\, \mathbf{out}(c, a).\mathbf{out}(c, a) \,\}$ and $Q = \{\, \mathbf{out}(c, a).\mathbf{out}(c, a), \mathbf{out}(c, a).\mathbf{out}(c, b) \,\}$. Clearly $P \sqsubseteq_{ct} Q$. Observe also that $Q \sqsubseteq_{ct} P$. This is because $\{w_1 \mapsto a, w_2 \mapsto b\} \sqsubseteq_s \{w_1 \mapsto a, w_2 \mapsto a\}$. Thus, $P \approx_{ct} Q$. But $P \not\approx_t Q$.

We show, however, that these two notions coincide for the class of *determinate processes.* In the context of the applied pi calculus determinate processes were previously studied by Cortier and Delaune in [23].

Definition 7. ***(Determinate process)*** *A ground process P is determinate if whenever $(P, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (T, \varphi)$ and $(P, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (T', \varphi')$ then $\varphi \approx_s \varphi'$.*

Intuitively, determinate processes are processes in which the adversary's static knowledge at any instance is completely determined by its past interaction with the protocol participants. Note that any ground trace is determinate.

As already mentioned above, it was demonstrated in [23] that trace equivalence coincides with observational equivalence for determinate processes. We show that $\approx_t$ and $\approx_{ct}$ also coincide for this class of processes.

Theorem 1. *If P and Q are ground processes then $P \approx_t Q$ implies $P \approx_{ct} Q$. Furthermore, if P and Q are determinate, then $P \approx_{ct} Q$ implies $P \approx_t Q$.*

We introduce a more fine-grained notion of trace equivalence, denoted $\approx_{ft}$.

Definition 8. *Given ground processes P and Q, we say that $P \sqsubseteq_{ft} Q$ whenever for all trace $T \in P$ there exists a trace $T' \in Q$ such that $T \approx_t T'$. We say that $P \approx_{ft} Q$ if $P \sqsubseteq_{ft} Q$ and $Q \sqsubseteq_{ft} P$.*

It follows directly form the definition that $\approx_{ft} \subset \approx_t$. The difference between these two relations is illustrated by the following example.

Example 6. Let P and Q be ground processes defined as follows:

$$\begin{aligned}
P = \{\ & \mathbf{out}(c, enc(a,k)).\mathbf{out}(c, enc(b,k)).\mathbf{in}(c,x).[x = enc(a,k)].\mathbf{out}(c,k),\\
& \mathbf{out}(c, enc(a,k)).\mathbf{out}(c, enc(b,k)).\mathbf{in}(c,x).[x = enc(b,k)].\mathbf{out}(c,k)\}\\
Q = \{\ & \mathbf{out}(c, enc(a,k)).\mathbf{out}(c, enc(b,k)).\mathbf{in}(c,x).[x = enc(dec(x,k),k)].\mathbf{out}(c,k)\}
\end{aligned}$$

where $k \in \mathcal{N}$ is a private name and a, b are constants. The test $x = enc(dec(x,k),k)$ simply checks whether x is an encryption with key k. It is not difficult to see that $P \approx_t Q$ but $P \not\approx_{ft} Q$.

Our procedure is able to check $\approx_{ct}$ (and hence $\approx_t$) for determinate processes. For non-determinate processes, we can check $\approx_{ft}$ and an over-approximation of $\approx_{ct}$ (see [16] for details) in order to under- and over-approximate $\approx_t$: as traces are determinate a procedure for checking $\approx_{ct}$ can be used to verify $\approx_{ft}$.

4 Modeling traces as Horn clauses

Our procedure is based on a fully abstract modeling of a trace into first-order Horn clauses. We give the details of this modeling; we start by giving some definitions that we need for defining the predicates used in the logic.

Symbolic labels and symbolic runs. We define the set of *symbolic labels* as

$$\mathsf{SLabels} = \{\mathbf{in}(c,t), \mathbf{out}(c), \mathbf{test} \mid t \in \mathsf{SMessages}, c \in \mathcal{C}\}$$

and the set of *symbolic runs* as the set of finite sequences of symbolic labels (see Figure 1). The empty sequence is denoted by ϵ. We will often be lazy and write (empty space) for ϵ. Intuitively, a symbolic label stands for a set of possible labels, and a symbolic run stands for a set of possible runs of the protocol.

Symbolic Recipes. We assume a set $\mathcal{Y}$ of *recipe variables* disjoint from $\mathcal{X}$. The set of terms $\mathcal{T}_{\mathcal{F},\mathcal{M},\mathcal{W},\mathcal{Y}}$ shall be called *symbolic recipes* and denoted by $\mathsf{SRecipes}$. We use capital letters X, Y, Z to range over $\mathcal{Y}$. Intuitively, a symbolic recipe stands for a set of recipes. We can extend the definition of substitutions to include variables from $\mathcal{Y}$ in its domain: we only consider substitutions that map variables in $\mathcal{Y}$ to $\mathsf{SRecipes}$. A ground substitution must map variables in $\mathcal{Y}$ to $\mathsf{Recipes}$. The notions of mgu and mgu_{R} is extended to symbolic recipes as expected.

Predicates. The predicates used in our modeling and the semantics of the predicates are given in Figure 1. The predicates are interpreted over a triple– a trace T, a frame φ and a substitution σ. We have four kinds of predicates, all of which have a symbolic run as an argument. Intuitively, the *reachability predicate* r_w says that each run represented by w is possible. The intruder knowledge predicate $\mathsf{k}_w(R, t)$ says that whenever a run represented by w happens, the (symbolic) message t can be constructed by the intruder using the (symbolic) recipe R. The identity predicate $\mathsf{i}_w(R, R')$ says that whenever the (symbolic) run SR happens, the (symbolic) recipes R and R' are recipes for the same (symbolic) term. The reachable identity predicate $\mathsf{ri}_w(R, R')$ is a short form for the conjunction of the predicates r_w and $\mathsf{i}_w(R, R')$.

Formulas and statements. We consider first-order formulas built using the above predicates and the usual connectives (conjunction, disjunction, negation, implication, existential and universal quantification). As in the case of predicates, a formula is interpreted over a triple consisting of a trace T, a frame φ and a substitution σ; and the semantics is defined as expected. For ground formulas we do not need the substitution σ and when a formula f is ground we simply write $(T, \varphi) \models f$ to denote that this formula holds for (T, φ). If moreover, $dom(\varphi) = \emptyset$, we simply write $T \models f$ for $(T, \emptyset) \models f$.

We now identify a subset of the formulas, which we shall call statements. Statements shall take the form of Horn clauses.

Definition 9. *A* statement *is a Horn clause of the form* $H \Leftarrow B_1, \ldots, B_n$ *where:*

1. $H \in \{\mathsf{r}_{\ell_1,\ldots,\ell_k}, \mathsf{k}_{\ell_1,\ldots,\ell_k}(R, t), \mathsf{i}_{\ell_1,\ldots,\ell_k}(R, R'), \mathsf{ri}_{\ell_1,\ldots,\ell_k}(R, R')\}$
2. *For each* $1 \leq i \leq n, B_i = \mathsf{k}_{\ell_1,\ldots,\ell_{j_i}}(X_i, t_i)$

for some $\ell_1, \ldots, \ell_k \in \mathsf{SLabels}$*,* $t \in \mathsf{SMessages}$*,* $R, R' \in \mathsf{SRecipes}$*,* $j_i \leq k$*,* $t_1, \ldots, t_n \in \mathsf{SMessages}$ *and* $X_1, \ldots, X_n \in \mathcal{Y}$*. Furthermore* $X_1, \ldots, X_n$ *are distinct variables and if* $H = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R, t)$ *then* $vars(t) \subseteq vars(t_1, \ldots, t_n)$*.*

As usual, we implicitly assume that in a Horn clause all variables are universally quantified. Hence, all statements are closed formulas.

The set of seed statements. Our procedure is based on a fully abstract modeling of a trace in first-order Horn clauses. In this section, given a trace T we define a set of statements $\mathsf{seed}(T)$ that serve as a starting point for the modeling. We also establish that $\mathsf{seed}(T)$ is a sound and (partially) complete abstraction of the trace T. In order to formally define $\mathsf{seed}(T)$, we start by fixing some conventions.

Symbolic Runs ($\ell \in \mathsf{SLabels}$):
$u, v, w := \epsilon \mid \ell, w$

Predicates ($w \in \mathsf{SRuns}, R \in \mathsf{SRecipes}, t \in \mathsf{SMessages}$):

r_w	(Reachability predicate)
$\mathsf{k}_w(R, t)$	(Intruder knowledge predicate)
$\mathsf{i}_w(R, R')$	(Identity predicate)
$\mathsf{ri}_w(R, R')$	(Reachable identity predicate)

Semantics ($\ell_i \in \mathsf{SLabels}, R \in \mathsf{SRecipes}, t \in \mathsf{SMessages}, T \in \mathsf{GndTraces}, \varphi \in \mathsf{Frames}$, σ a ground substitution):

$(T, \varphi_0, \sigma) \models \mathsf{r}_{\ell_1,\ldots,\ell_i}$ if $(T, \varphi_0) \xrightarrow{L_1} (T_1, \varphi_1) \xrightarrow{L_2} \ldots \xrightarrow{L_n} (T_n, \varphi_n)$ such that $\ell_i\sigma =_{\mathsf{R}} L_i\varphi_{i-1}$ for all $1 \leq i \leq n$

$(T, \varphi_0, \sigma) \models \mathsf{k}_{\ell_1,\ldots,\ell_i}(R, t)$ if when $(T, \varphi_0) \xrightarrow{L_1} (T_1, \varphi_1) \xrightarrow{L_2} \ldots \xrightarrow{L_n} (T_n, \varphi_n)$ such that $\ell_i\sigma =_{\mathsf{R}} L_i\varphi_{i-1}$ for all $1 \leq i \leq n$ then $\varphi_n \vdash^{R\sigma} t\sigma$

$(T, \varphi_0, \sigma) \models \mathsf{i}_{\ell_1,\ldots,\ell_i}(R, R')$ if there exists t s.t. $(T, \varphi_0, \sigma) \models \mathsf{k}_{\ell_1,\ldots,\ell_i}(R, t)$ and $(T, \varphi_0, \sigma) \models \mathsf{k}_{\ell_1,\ldots,\ell_i}(R', t)$

$(T, \varphi_0, \sigma) \models \mathsf{ri}_{\ell_1,\ldots,\ell_i}(R, R')$ if $(T, \varphi_0, \sigma) \models \mathsf{r}_{\ell_1,\ldots,\ell_i}$ and $(T, \varphi_0, \sigma) \models \mathsf{i}_{\ell_1,\ldots,\ell_i}(R, R')$

Fig. 1: Predicates

Let $T = a_1.a_2.\ldots.a_n$ be a ground trace. We assume the following naming conventions: *(i)* if a_i is a receive action then $a_i = \mathbf{in}(c_i, x_i)$; *(ii)* $x_i \neq x_j$ for any $i \neq j$; *(iii)* if a_i is a send action then $a_i = \mathbf{out}(c_i, t_i)$; *(iv)* if a_i is a test action then $a_i = [s_i \overset{?}{=} t_i]$. Moreover, for each $1 \leq i \leq n$, let $\ell_i \in \mathsf{SLabels}$ be as follows:

$$\ell_i = \begin{cases} \mathbf{in}(c_i, x_i) & \text{if } a_i = \mathbf{in}(c_i, x_i) \\ \mathbf{out}(c_i) & \text{if } a_i = \mathbf{out}(c_i, t_i) \\ \mathbf{test} & \text{if } a_i = [s_i \overset{?}{=} t_i] \end{cases} .$$

For each $0 \leq m \leq n$, let the sets $R(m)$, $S(m)$ and $T(m)$ respectively denote the indices of the receive actions, send actions and test actions amongst $a_1, \ldots, a_m$. Formally, $R(m) = \{i \mid 1 \leq i \leq m, a_i = \mathbf{in}(c_i, x_i)\}$, $S(m) = \{i \mid 1 \leq i \leq m, a_i = \mathbf{out}(c_i, t_i)\}$ and $T(m) = \{i \mid 1 \leq i \leq m, a_i = [s_i \overset{?}{=} t_i]\}$ Given a set of public names $\mathcal{M}_0 \subseteq \mathcal{M}$, *set of seed statements* associated to T and $\mathcal{M}_0$, denoted $\mathsf{seed}(T, \mathcal{M}_0)$, is defined to be the set of statements given in Figure 2. If $\mathcal{M}_0 = \mathcal{M}$, then $\mathsf{seed}(T, \mathcal{M})$ is said to be the set of seed statements associated to T and in this case we write $\mathsf{seed}(T)$ as a shortcut for $\mathsf{seed}(T, \mathcal{M})$. While constructing $\mathsf{seed}(T, \mathcal{M})$, we apply $\mathrm{mgu}_{\mathsf{R}}$ to all tests. In addition, we also apply finite variants. This allows us to *get rid* of rewriting in our procedure.

For a set of statements K, we denote by $\mathcal{H}(K)$ the least Herbrand model of K $\cup\{\mathsf{k}_{\ell_1,\ldots,\ell_{n+1}}(X, x) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_n}(X, x)\}_{n\in\mathbb{N}} \cup \{\mathsf{i}_{\ell_1,\ldots,\ell_{n+1}}(X_1, X_2) \Leftarrow \mathsf{i}_{\ell_1,\ldots,\ell_n}(X_1, X_2)\}_{n\in\mathbb{N}}$. We show that as far as reachability predicates and intruder knowledge predicates are concerned, the set $\mathsf{seed}(T)$ is a complete abstraction T.

$\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow} \Leftarrow \{\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow)\}_{j\in R(m)}$
  for all $0 \leq m \leq n$
  for all $\sigma \in \mathrm{mgu}_{\mathsf{R}}(\{s_k = t_k\}_{k\in T(m)})$
  for all $\tau \in \mathrm{variants}(\ell_1\sigma, \ldots, \ell_m\sigma)$

$\mathsf{k}_{\ell_1\tau\downarrow,\ldots,\ell_m\tau\downarrow}(w_{|S(m)|}, t_m\tau\downarrow) \Leftarrow \{\mathsf{k}_{\ell_1\tau\downarrow,\ldots,\ell_{j-1}\tau\downarrow}(X_j, x_j\tau\downarrow)\}_{j\in R(m)}$
  for all $m \in S(n)$
  for all $\tau \in \mathrm{variants}(\ell_1, \ldots, \ell_m, t_m)$

$\mathsf{k}(c, c) \Leftarrow$
  for all public names $c \in \mathcal{M}_0$

$\mathsf{k}_{\ell_1,\ldots,\ell_m}(f(Y_1, \ldots, Y_k), f(y_1, \ldots, y_k)\tau\downarrow) \Leftarrow \{\mathsf{k}_{\ell_1,\ldots,\ell_m}(Y_j, y_j\tau\downarrow)\}_{j\in\{1,\ldots,k\}}$
  for all $0 \leq m \leq n$
  for all function symbols f of arity k
  for all $\tau \in \mathrm{variants}(f(y_1, \ldots, y_k))$.

Fig. 2: Seed statements

Theorem 2. *Let T be a ground trace.*

- *(Soundness.) For any $f \in \mathsf{seed}(T) \cup \mathcal{H}(\mathsf{seed}(T))$ we have that $T \models f$.*
- *(Completeness.) If $(T, \emptyset) \xrightarrow{L_1,\ldots,L_m} (S, \varphi)$ then (i) $\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_m\varphi\downarrow} \in \mathcal{H}(\mathsf{seed}(T))$, and (ii) if $\varphi \vdash^R t$ then $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_m\varphi\downarrow}(R, t\downarrow) \in \mathcal{H}(\mathsf{seed}(T))$.*

Remark 2. Note that the set $\mathsf{seed}(T)$ is only partially complete as we have not shown above that if $\varphi \vdash^R t$ and $\varphi \vdash^{R'} t$ then $\mathsf{i}_{L_1\varphi\downarrow,\ldots,L_m\varphi\downarrow} \in \mathcal{H}(\mathsf{seed}(T))$. We will shortly show how the completeness of $\mathsf{seed}(T)$ can be built upon to achieve a) full abstraction of T and b) procedures for checking equivalences $\approx_{ct}$ and $\sqsubseteq_{ft}$.

5 Procedure for deciding trace equivalence

We now present a procedure for verifying trace equivalence. At a high level, this consists of the following two steps that we will detail later.

1. A saturation procedure which constructs a set of *simple* statements from the set $\mathsf{seed}(T)$ which we will call *solved* statements. The saturation procedure ensures that the set of solved statements is a complete abstraction of T.
2. Given two ground processes P and Q, we saturate the set of seed statements for traces of P and Q and then use the solved statements to decide whether P and Q are trace equivalent.

5.1 Knowledge bases and saturation

The saturation procedure manipulates a set of statements called a knowledge base.

$$\textsc{Resolution}\ \frac{\begin{array}{c} f \in K, g \in K_{\mathsf{solved}}, \\ f = \Big(H \Leftarrow \mathsf{k}_{uv}(X,t), B_1, \ldots, B_n\Big) \qquad g = \Big(\mathsf{k}_w(R,t') \Leftarrow B_{n+1}, \ldots, B_m\Big) \\ \sigma = \mathrm{mgu}(\mathsf{k}_u(X,t), \mathsf{k}_w(R,t')) \qquad t \notin \mathcal{X} \end{array}}{K = K \oplus h \text{ where } h = \Big((H \Leftarrow B_1, \ldots, B_m)\sigma\Big)}$$

$$\textsc{Equation}\ \frac{\begin{array}{c} f, g \in K_{\mathsf{solved}}, \qquad f = \Big(\mathsf{k}_u(R,t) \Leftarrow B_1, \ldots, B_n\Big) \\ g = \Big(\mathsf{k}_{u'v'}(R',t') \Leftarrow B_{n+1}, \ldots, B_m\Big) \qquad \sigma = \mathrm{mgu}(\mathsf{k}_u(\_,t), \mathsf{k}_{u'}(\_,t')) \end{array}}{K = K \oplus h \text{ where } h = \Big((\mathsf{i}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)\sigma\Big)}$$

$$\textsc{Test}\ \frac{\begin{array}{c} f, g \in K_{\mathsf{solved}}, \qquad f = \Big(\mathsf{i}_u(R,R') \Leftarrow B_1, \ldots, B_n\Big) \\ g = \Big(\mathsf{r}_{u'v'} \Leftarrow B_{n+1}, \ldots, B_m\Big) \qquad \sigma = \mathrm{mgu}(u,u') \end{array}}{K = K \oplus h \text{ where } h = \Big((\mathsf{ri}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)\sigma\Big)}$$

Fig. 3: Saturation rules

Definition 10. *Given a statement* $f = H \Leftarrow B_1, \ldots, B_n$,

- f *is said to be* solved *if for all* $1 \leq i \leq n$, $B_i = \mathsf{k}_{\ell_1,\ldots,\ell_{j_i}}(X_i, x_i)$ *for some variables* $x_i \in \mathcal{X}, X_i \in \mathcal{Y}$.
- f *is said to be* well-formed *if whenever it is solved and* $H = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t)$, *we have that* $t \notin \mathcal{X}$.

A set of well-formed *statements is called a* knowledge base. *If* K *is a knowledge base, we define* $K_{\mathsf{solved}} = \{f \in K \mid f \text{ is solved }\}$ *to be the knowledge base restricted to the solved statements.*

Given an initial knowledge base K, the saturation procedure produces another knowledge base $\mathsf{sat}(K)$ as follows. First, new statements are *generated*. Then the knowledge base is *updated* with the new statements. This two-step process continues until a fixed-point is achieved. We describe the two steps in the procedure.

Generating new statements. Given a knowledge base K, new statements f are generated by applying the rules in Figure 3.

Update. The first step while updating the knowledge base by f is to convert f into a canonical form.

Definition 11. *Given a solved deduction statement* f, *we define its* canonical form *to be the statement* $f\Downarrow$ *obtained by first applying Rule* Rename *as many times as possible and then applying Rule* Remove *as many times as possible:*

$$\textsc{Rename}\ \frac{H \Leftarrow \mathsf{k}_u(X,x), \mathsf{k}_{uv}(Y,x), B_1, \ldots, B_n}{(H \Leftarrow \mathsf{k}_u(X,x), B_1, \ldots, B_n)\{Y \mapsto X\}}$$

$$\textsc{Remove}\ \frac{H \Leftarrow \mathsf{k}_u(X,x), B_1, \ldots, B_n \qquad x \notin vars(H)}{H \Leftarrow B_1, \ldots, B_n}$$

For any other type of statement, the canonical form $f{\Downarrow}$ is defined to be f.

It is easy to see that any fact f can be converted into a canonical form. After a canonical form has been obtained, we perform another check before $f{\Downarrow}$ can be added to the knowledge base. Intuitively, this check ensures that we add enough identity predicates in the knowledge base. We need the following definition for the update rule.

Definition 12. *The set of* consequences *of a knowledge base K, denoted $\mathbf{cons}(K)$, is the smallest set such that:*

$$\text{AXIOM}\ \frac{}{\mathsf{k}_{uv}(R,t) \Leftarrow \mathsf{k}_{u}(R,t), B_1, \ldots, B_m \in \mathbf{cons}(K)}$$

$$\text{RES}\ \frac{\begin{array}{c} H \Leftarrow B_1, \ldots, B_n \in K \qquad \sigma \textit{ a substitution} \\ B_1\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{cons}(K), \ldots, B_n\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{cons}(K) \end{array}}{H\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{cons}(K)}$$

Given a knowledge base K and a statement f, the *update of K by f*, denoted $K \oplus f$, is defined to be $K \cup \{f{\Downarrow}\}$ if the head of f is not of the form $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t)$. Otherwise, let

$$f{\Downarrow} = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,t_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,t_n)$$

and $K \oplus f =$

- $K \cup \{f{\Downarrow}\}$ if f is solved and for any R' we have that $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,t_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,t_n) \notin \mathbf{cons}(K_{\mathsf{solved}})$.
- $K \cup \{\mathsf{i}_{\ell_1,\ldots,\ell_k}(R,R') \Leftarrow \{\mathsf{k}_{\ell_1,\ldots,\ell_{i_j}}(X_j,t_j)\}_{j\in\{1,\ldots,n\}}\}$ if f is solved and R' is such that $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,t_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,t_n) \in \mathbf{cons}(K_{\mathsf{solved}})$.
- $K \cup \{f{\Downarrow}\}$ if f is not solved.

Note that update is not a function, namely that there may be several $R', i_1, \ldots, i_n$ such that $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,t_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,t_n) \in \mathbf{cons}(K_{\mathsf{solved}})$. However, we need to compute only one such R'.

Initial knowledge base. One question that naturally arises is what is the initial knowledge base for the saturation procedure. Given a ground trace T, the initial knowledge base for the saturation procedure is defined as follows.

Definition 13. *Given a set of statements S, the* initial knowledge base *associated to S, denoted $K_i(S)$, is defined to be the empty knowledge base updated by the set S, i.e., $K_i(S) = \emptyset \oplus_{f\in S} f$. If T is a ground trace, we write $K_i(T)$ for $K_i(\mathsf{seed}(T))$.*

Observe that $K_i(T)$ depends on the order in which statements in $\mathsf{seed}(T)$ are updated. The exact order, however, is not important and our results hold regardless of the order chosen. The saturation procedure takes $K_i(T)$ as an input and produces a knowledge base $\mathsf{sat}(K_i(T))$. The reason for choosing $K_i(T)$ instead of $\mathsf{seed}(T)$ as the starting point of the saturation procedure is that $\mathsf{seed}(T)$ may not be a knowledge base, i.e., may contain non well-formed statements. The set $K_i(T)$ is, however, a knowledge base.

Proposition 1. *Given a ground trace T, the set $K_i(T)$ is a knowledge base.*

Soundness and completeness of the saturation procedure. We shall now show that the set of solved statements in $\mathsf{sat}(K_i(T))$ is a sound and complete abstraction of a ground trace T. Given a set of statements K we denote by $\mathcal{H}_{\mathsf{e}}(K)$ the smallest set of ground terms such that

– $\mathcal{H}(K) \subseteq \mathcal{H}_{\mathsf{e}}(K)$,
– $\mathcal{H}_{\mathsf{e}}(K)$ is closed under congruence rules for each $\mathsf{i}_w(R, R') \in \mathcal{H}_{\mathsf{e}}(K)$, and
– i_w is monotonic in w, i.e., $\mathsf{i}_u(R, R') \in \mathcal{H}_{\mathsf{e}}(K)$ implies $\mathsf{i}_{uv}(R, R') \in \mathcal{H}_{\mathsf{e}}(K)$.

A formal definition is given in [16].

Theorem 3. *Let T be a ground trace and let $K = \mathsf{sat}(K_i(T))$.*

– *(Soundness.) For any $f \in K \cup \mathcal{H}_{\mathsf{e}}(K)$ we have $T \models f$.*
– *(Completeness.) If $(T, \emptyset) \xrightarrow{L_1,\ldots,L_n} (S, \varphi)$ then (i) $\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow} \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$, (ii) if $\varphi \vdash^R t$ then $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, t\downarrow) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$, and (iii) if $\varphi \vdash^R t$ and $\varphi \vdash^{R'} t$, then $\mathsf{i}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, R') \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$.*

Effectiveness of the saturation procedure. We have shown that the set of solved statements in $\mathsf{sat}(K_i(T))$ form a sound and complete abstraction for the trace T. However this set is infinite and may not be effectively computable. This may be because of following reasons.

– The set $\mathsf{seed}(T)$ for a ground trace T is infinite. Hence the saturation procedure may continue forever. We will, however, shortly show that for the saturation procedure we only need to consider the saturation of the set $K_i(\mathsf{seed}(T, \mathcal{M}_0))$ where $\mathcal{M}_0$ is the set of public names occurring in T (see Lemma 1). The set $\mathsf{sat}(K_i(T))$ can then be computed from this set. Since the set $K_i(\mathsf{seed}(T, \mathcal{M}_0))$ is finite, this means that all intermediate knowledge bases in the saturation procedure are finite.
– For the update rule, we have to check that given a knowledge base K, term t, labels $\ell_1, \ldots, \ell_k$, indices $1 \leq i_1, \ldots i_n \leq k$, variables $x_1, \ldots, x_n \in \mathcal{X}$ and recipe variables $X_1, \ldots, X_n \in \mathcal{Y}$, whether

$$\exists R.\ \mathsf{k}_{\ell_1,\ldots,\ell_k}(R, t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n) \in \mathbf{cons}(K_{\mathsf{solved}}).$$

Furthermore, if the check succeeds then we have to compute one such R. We will show that can be achieved if K is finite (see Lemma 2).
– The saturation procedure may itself not terminate even if the initial knowledge base is finite. As pointed out in the Introduction, we conjecture that the saturation procedure terminates for subterm convergent rewrite systems, but were unable to show the termination.

The following lemma allows us to compute the $\mathsf{sat}(K_i(T))$ from the set $\mathsf{sat}(K_i(\mathit{seed}(M_0, T)))$ where M_0 is the set of public names occurring in T.

Lemma 1. *Let T be a ground trace and $M_T \subseteq \mathcal{M}$ be the public names occurring in T. Let $K_{\mathcal{M}} = \{\{\mathsf{k}(m, m) \Leftarrow\}_{m\in\mathcal{M}} \cup \{\mathsf{i}(m, m) \Leftarrow\}_{m\in\mathcal{M}} \cup \{\mathsf{ri}(m, m) \Leftarrow\}_{m\in\mathcal{M}}\}$. Then $\mathsf{sat}(K_i(T)) = \mathsf{sat}(K_i(\mathit{seed}(M_T, T))) \cup K_{\mathcal{M}}$.*

The following lemma implies that the update step terminates if we only have a finite number of solved statements in the knowledge base.

Lemma 2. *Given a finite set of statements K, term t, labels $\ell_1, \ldots, \ell_k$, indices $1 \leq i_1, \ldots i_n \leq k$, variables $x_1, \ldots, x_n \in \mathcal{X}$ and recipe variables $X_1, \ldots, X_n \in \mathcal{Y}$, it is decidable if there is an R such that $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R, t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1)$, $\ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n) \in \mathbf{cons}(K_{\mathsf{solved}})$. If the answer to the decision procedure is "Yes", then we can compute one such R.*

5.2 Algorithm for checking equivalence

Once we constructed saturated knowledge bases for the seed statements for ground determinate processes P_0 and P_1, we can check trace equivalence $\approx_{ct}$. The algorithm for checking $\approx_{ct}$ for determinate processes, automatically gives an algorithm for checking $\approx_{ft}$ for non-determinate processes. It suffices to check for $T \sqsubseteq_{ct} P$ for a ground trace T and ground determinate process P. This basically involves checking two tests which are summarized in Figure 4. We briefly describe them below.

– Reach checks whether all sequence of actions executable by T are also executable by P. To do this, we carry out the following operations for *each* statement $\left.\mathsf{r}_{l_1,\ldots,l_n} \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i\in\{1,\ldots,m\}}\right) \in \{\mathsf{sat}(\mathsf{seed}(T))\}_{\mathsf{solved}}$. *(a)* First we pick fresh constants $c_1, \ldots, c_k$ for each of the variables occurring in $l_1, \ldots, l_n$ and fix a bijection σ between them. *(b)* Next for each $1 \leq i \leq n$ s.t. l_i is $\mathbf{in}(d_i, t_i)$, we construct *one* recipe R_i such that $\mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(\{\mathsf{sat}(\mathsf{seed}(T))\}_{\mathsf{solved}})$. Such an R_i exists thanks to the completeness of the saturation procedure. We let $M_i = \mathbf{in}(d_i, R_i)$. *(c)* For each $1 \leq i \leq n$ s.t. $l_i = \mathbf{test}$ or $\mathbf{out}(d_i)$ we let $M_i = l_i$. *(d)* We check if $(P, \emptyset) \xRightarrow{M_1,\ldots,M_n} (T', \varphi)$. If all the Reach tests pass then we go to test Identity. Otherwise we declare T to be not trace-contained in P.
– The test Identity checks that all the equality tests that hold after an execution of T hold after a similar execution in P. In order to do this, we carry out the following operations for *each* statement $\mathsf{ri}_{l_1,\ldots,l_n}(R, R') \Leftarrow \left.\{\mathsf{k}_{w_i}(X_i, x_i)\}_{i\in\{1,\ldots,m\}}\right) \in \{\mathsf{sat}(\mathsf{seed}(T))\}_{\mathsf{solved}}$. We construct $M_1, \ldots, M_n$ as in the Reach test and check if there is a T' such that $(P, \emptyset) \xRightarrow{M_1,\ldots,M_n} (T', \varphi)$ and the recipes $R\{X_i \mapsto x_i\sigma\}$ and $R'\{X_i \mapsto x_i\sigma\}$ are equal in frame φ.

Note that performing the tests requires deciding if, given t, and w, $\mathsf{k}_w(R, t) \in \mathcal{H}(K)$ for some recipe R for a knowledge base K containing only solved statements. This is similar to checking if $\left(\mathsf{k}_w(R, t) \Leftarrow\right) \in \mathbf{cons}(K)$.

Theorem 4. *Let T be a ground trace and let P be a ground determinate process. Let K be the set of solved statements from a saturated knowledge base associated to T. Then $T \sqsubseteq_{ct} P$ iff all the tests in Figure 4 hold.*

$$\textsc{Reach}\ \frac{\begin{array}{c} \left(\mathsf{r}_{l_1,\dots,l_n} \Leftarrow \{\mathsf{k}_{w_i}(X_i,x_i)\}_{i\in\{1,\dots,m\}}\right) \in \{\mathsf{sat}(\mathsf{seed}(T))\}_{\mathsf{solved}} \\ c_1,\dots,c_k \text{ fresh constants} \\ \sigma : \mathit{vars}(l_1,\dots,l_n) \to \{c_1,\dots,c_k\} \text{ is a bijection} \\ \mathsf{k}_{l_1\sigma,\dots,l_{i-1}\sigma}(R_i,t_i\sigma) \in \mathcal{H}(\{\mathsf{sat}(\mathsf{seed}(T))\}_{\mathsf{solved}}) \text{ for all } i \text{ s.t. } l_i = \mathbf{in}(d_i,t_i) \\ M_i = l_i \text{ if } l_i \in \{\mathbf{test},\mathbf{out}(\_)\} \qquad M_i = \mathbf{in}(d_i,R_i) \text{ if } l_i = \mathbf{in}(d_i,t_i)\end{array}}{(P,\emptyset) \xRightarrow{M_1,\dots,M_n} (T',\varphi)}$$

$$\textsc{Identity}\ \frac{\begin{array}{c} \left(\mathsf{ri}_{l_1,\dots,l_n}(R,R') \Leftarrow \{\mathsf{k}_{w_i}(X_i,x_i)\}_{i\in\{1,\dots,m\}}\right) \in \{\mathsf{sat}(\mathsf{seed}(T))\}_{\mathsf{solved}} \\ c_1,\dots,c_k \text{ fresh constants} \\ \sigma : \mathit{vars}(l_1,\dots,l_n) \to \{c_1,\dots,c_k\} \text{ is a bijection} \\ \mathsf{k}_{l_1\sigma,\dots,l_{i-1}\sigma}(R_i,t_i\sigma) \in \mathcal{H}(\{\mathsf{sat}(\mathsf{seed}(T))\}_{\mathsf{solved}}) \text{ for all } i \text{ s.t. } l_i = \mathbf{in}(t_i) \\ M_i = l_i \text{ if } l_i \in \{\mathbf{test},\mathbf{out}(\_)\} \qquad M_i = \mathbf{in}(d_i,R_i) \text{ if } l_i = \mathbf{in}(d_i,t_i)\end{array}}{(P,\emptyset) \xRightarrow{M_1,\dots,M_n} (T',\varphi) \text{ such that } (R\omega = R'\omega)\varphi \text{ where } \omega = \{X_i \mapsto x_i\sigma\}}$$

Fig. 4: Tests for checking trace inclusion

6 Prototype and case studies

We implemented the procedure for checking equivalence in a prototype, AKiSs (Active Knowledge In Security protocols). AKiSs is written in OCaml and has about 2000 lines of source code, including code for computing complete sets of finite variants and complete sets of equational unifiers. For protocol specification, we allow for an operator *interleave* which models parallel composition of processes and an operator *sequence* for modeling protocols structured in phases.

We used AKiSs to verify the equivalences in Examples 5 and 6. Using AKiSs we were able to verify strong secrecy for Denning-Sacco-Blanchet [11] and Needham-Schroeder-Lowe (NSL) [36], resistance to guessing attacks in the EKE protocol [9], and, more interestingly, anonymity of the FOO [32] and Okamoto [38] electronic voting protocols.[3] To our knowledge, AKiSs is the only tool that can verify FOO and Okamoto automatically. We briefly discuss the salient points of these examples below. AKiSs along with all the discussed examples is available on: `http://www.lsv.ens-cachan.fr/~ciobaca/akiss/`. Details of the modeling can also be found in [16].

[3] Please note that as defined in [38], modeling of Okamoto's protocol requires private channels. As we do not have private channels in our calculus, we transform the protocol so that every message sent by honest participants on a private channel is sent encrypted under a key not known to the adversary

Strong flavors of confidentiality. The *strong secrecy* property was introduced by Blanchet in [11] and we rephrase it here in our setting. Let P be a protocol with x as the only free variable of P. Then x is said to be *strongly secret* if

$$\mathbf{in}(c, x_1).\mathbf{in}(c, x_2).(P\{x \mapsto x_1\}) \approx_t \mathbf{in}(c, x_1).\mathbf{in}(c, x_2).(P\{x \mapsto x_2\}).$$

Intuitively, the attacker cannot distinguish the processes using variables x_1 and x_2 even though it can choose arbitrary (public) values for these variables. The definition generalizes to multiple variables in the expected way. We illustrate this property on a Denning-Sacco-Blanchet protocol. Informally, the protocol can be described as follows.

$$\begin{array}{l}\mathsf{A} \to \mathsf{B} : \mathsf{aenc}(\mathsf{sign}(\mathsf{pair}(\mathsf{pk}(ska), \mathsf{pair}(\mathsf{pk}(skb, k))), ska), \mathsf{pk}(skb)) \\ \mathsf{B} \to \mathsf{A} : \mathsf{enc}(x, k)\end{array}$$

A sends to B a fresh symmetric session key k together with A's and B's public keys. This is signed with A's secret key and (asymmetrically) encrypted with B's public key. Upon receiving this message, B decrypts it, checks the signature and uses the fresh session key to symmetrically encrypt a secret x. We used AKiSs to verify this protocol for strong secrecy of x (with one session of A and B). This protocol is determinate, and hence we used $\approx_{ct}$ to verify the protocol. The verification succeeds as expected.

A variant of the protocol [11] consists in letting A also send out a secret y encrypted with k changing the first message to

$$\mathsf{A} \to \mathsf{B} : \mathsf{pair}(\mathsf{aenc}(\mathsf{sign}(\mathsf{pair}(\mathsf{pk}(ska), \mathsf{pair}(\mathsf{pk}(skb, k))), ska), \mathsf{pk}(skb)), \mathsf{enc}(y, k))$$

In this case the protocol does not respect strong secrecy of x, y as, by choosing $x_1 = y_1$ and $x_2 \neq y_2$, the attacker can distinguish the two situations by testing the equality of the encryptions of x and y. This attack is again found by AKiSs. AKiSs also verifies strong secrecy of the nonce generated by the responder in the Needham-Schroeder-Lowe (NSL) [36] protocol. Once again, the modeling of NSL leads to determinate processes, and we used $\approx_{ct}$ for our verification.

We also used AKiSs to verify the above protocols for *real-or-random* secrecy. This property is useful to model resistance to offline guessing attacks in password protocols [8]. We show that the EKE protocol [9] is resistant to offline guessing attacks. As EKE also leads to determinate processes, we used the $\approx_{ct}$ relation.

Anonymity for electronic voting protocol. A voting protocol must respect voter privacy: the adversary should not be able to learn how each voter voted. AKiSs can automatically verify voter privacy in the FOO electronic voting protocol [32] and the Okamoto protocol [38]. Voter privacy is naturally modeled as an equivalence property [26, 7]: it is not possible to distinguish the situation where honest voter A votes 'yes' and honest B votes 'no' from the situation that A votes 'no' and B votes 'yes'. Note that our modeling of the protocols is exactly the same as in [26]. We assume that *only* voters A and B are honest while all other entities are dishonest. An arbitrary number of dishonest voters are however subsumed by the attacker and need not be modeled directly. Both the protocols do not lead to determinate processes. Therefore, we proved the relation $\approx_{ft}$. To our knowledge, no other tool can handle this automatically. We are aware of two other attempts for verifying the FOO protocol. Using ProVerif [11], Delaune *et al.* [28], verify a transformation of the protocol. However, the soundness of this transformation has never been proven. Chothia *et al.* [20] verify a different notion of anonymity (also based on process equivalence) using the μCRL tool. However, the attacker they consider is only an observer that cannot interact with the protocol participants, yielding a finite state system.

Efficiency. On a standard modern laptop, AKiSs takes a few minutes (e.g. 3 mins for FOO) to carry out the above verification. The use of a multi-core server already reduces these timings by about 40%. We expect that some optimizations of the saturation procedure and the use of more efficient data structures will diminish these times significantly. Most of the computational effort goes into the saturation of the traces. Interleaving individual roles of a protocol introduces an exponential blowup on the number of traces and saturations to perform. However, it would be straightforward to scale to larger protocols and more sessions by parallelizing the saturation of these traces (e.g. on clusters of machines).

7 Conclusion and future work

We present a novel Horn-clause resolution based procedure for verifying equivalence properties for a bounded number of sessions of cryptographic protocols. This approach is validated by implementing it in the tool AKiSs, and we are able to handle examples which are out of the scope of existing tools.

There are several directions for future work. The implementation of the tool should be optimized and more examples from electronic voting, RFID protocols and auction protocols which all have requirements stated in terms of equivalences should be analyzed. We would also like to take disequalities into account. It will allow to verify processes with else branches, important in a number of practical examples, e.g., passport protocols discussed in [5]. Another direction would be to extend the procedure to allow AC (Associative/Commutative) operators in order to treat protocols based on exclusive-or or Diffie-Hellman exponentiations.

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 *28th Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM Press, 2001.
3. M. Abadi and C. Fournet. Private authentication. *Theoretical Computer Science*, 322(3):427–476, 2004.
4. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. Comput.*, 148(1):1–70, 1999.

5. M. Arapinis, T. Chothia, E. Ritter, and M. D. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Comp. Soc. Press, 2010.
6. A. Armando et al. The AVISPA tool for the automated validation of internet security protocols and applications. In *17th International Conference on Computer Aided Verification (CAV'05)*, LNCS, pages 281–285. Springer, 2005.
7. M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *21st Computer Security Foundations Symposium (CSF'08)*. IEEE Comp. Soc. Press, 2008.
8. M. Baudet. Deciding security of protocols against off-line guessing attacks. In *12th Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM Press, 2005.
9. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Symposium on Security and Privacy (S&P'92)*, pages 72–84. IEEE Comp. Soc. Press, 1992.
10. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Comp. Soc. Press, 2001.
11. B. Blanchet. Automatic proof of strong secrecy for security protocols. In *Symposium on Security and Privacy (S&P'04)*, pages 86–100, 2004.
12. B. Blanchet, M. Abadi, and C. Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *Symposium on Logic in Computer Science*, pages 331–340. IEEE Comp. Soc. Press, 2005.
13. J. Borgström. *Equivalences and Calculi for Formal Verifiation of Cryptographic Protocols*. Phd thesis, EPFL, Switzerland, 2008.
14. J. Borgström, S. Briais, and U. Nestmann. Symbolic bisimulation in the spi calculus. In *15th Int. Conference on Concurrency Theory (CONCUR'04)*, volume 3170 of *LNCS*, pages 161–176. Springer, 2004.
15. M. Bruso, K. Chatzikokolakis, and J. den Hartog. Analysing unlinkability and anonymity using the applied pi calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Comp. Soc. Press, 2010.
16. R. Chadha, Ş. Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. Technical report, Oct. 2011. `http://hal.inria.fr/inria-00632564/en/`.
17. V. Cheval, H. Comon-Lundh, and S. Delaune. Automating security analysis: symbolic equivalence of constraint systems. In *International Joint Conference on Automated Reasoning (IJCAR'10)*, LNAI, pages 412–426. Springer, 2010.
18. V. Cheval, H. Comon-Lundh, and S. Delaune. Trace equivalence decision: Negative tests and non-determinism. In *18th Conference on Computer and Communications Security (CCS'11)*, pages 321–330. ACM Press, 2011.
19. Y. Chevalier and M. Rusinowitch. Decidability of equivalence of symbolic derivations. *Journal of Automated Reasoning*, 2010. To appear.
20. T. Chothia, S. Orzan, J. Pang, and M. Torabi Dashti. A framework for automatically checking anonymity with *mu* crl. In *2nd Symposium on Trustworthy Global Computing (TGC'06)*, volume 4661 of *LNCS*, pages 301–318. Springer, 2007.
21. Ş. Ciobâcă. Computing finite variants for subterm convergent rewrite systems. Research Report LSV-11-06, LSV, ENS Cachan, France, 2011.
22. H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In *16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *LNCS*, pages 294–307. Springer, 2005.

23. V. Cortier and S. Delaune. A method for proving observational equivalence. In *22nd Computer Security Foundations Symposium (CSF'09)*, pages 266–276. IEEE Comp. Soc. Press, 2009.
24. M. Dahl, S. Delaune, and G. Steel. Formal analysis of privacy for vehicular mix-zones. In *15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *LNCS*, pages 55–70. Springer, 2010.
25. S. Delaune, S. Kremer, and O. Pereira. Simulation based security in the applied pi calculus. In *29th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'09)*, volume 4 of *Leibniz International Proceedings in Informatics*, pages 169–180. Leibniz-Zentrum für Informatik, 2009.
26. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.
27. 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.
28. S. Delaune, M. D. Ryan, and B. Smyth. Automatic verification of privacy properties in the applied pi-calculus. In *2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security (IFIPTM'08)*, volume 263 of *IFIP Conference Proceedings*, pages 263–278. Springer, 2008.
29. D. Dolev and A. Yao. On the security of public key protocols. In *22nd Symposium on Foundations of Computer Science (FOCS'81)*, pages 350–357. IEEE Comp. Soc. Press, 1981.
30. L. Durante, R. Sisto, and A. Valenzano. Automatic testing equivalence verification of spi calculus specifications. *ACM Transactions on Software Engineering and Methodology*, 12(2):222–284, 2003.
31. S. Escobar, C. Meadows, and J. Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design V*, volume 5705 of *LNCS*, pages 1–50. Springer, 2009.
32. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology — AUSCRYPT '92*, volume 718 of *LNCS*, pages 244–251. Springer, 1992.
33. J. Goubault-Larrecq. Deciding $\mathcal{H}_1$ by resolution. *Information Processing Letters*, 95(3):401–408, 2005.
34. H. Hüttel. Deciding framed bisimilarity. In *4th International Workshop on Verification of Infinite-State Systems (INFINITY'02)*, pages 1–20, 2002.
35. J. Liu and H. Lin. A complete symbolic bisimulation for full applied pi calculus. In *36th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'10)*, volume 5901 of *LNCS*, pages 552–563. Springer, 2010.
36. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer, 1996.
37. P. Narendran, F. Pfenning, and R. Statman. On the unification problem for cartesian closed categories. *J. Symb. Log.*, 62(2):636–647, 1997.
38. T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *5th Int. Security Protocols Workshop*, volume 1361 of *LNCS*, pages 25–35. Springer, 1997.
39. A. Tiu and J. Dawson. Automating open bisimulation checking for the spi-calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 307–321. IEEE Comp. Soc. Press, 2010.
40. C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *16th International Conference on Automated Deduction (CADE'99)*, volume 1632 of *LNCS*, pages 314–328. Springer, 1999.