Chapter 4

Deducibility Constraints

In this chapter, we present the NP-complete decision procedure for a bounded number of sessions by H. Comon-Lundh *et al.* [113]. In this setting (*i.e.* finite number of sessions), deducibility constraint systems have become the standard model for verifying security properties, with a special focus on secrecy. Starting with a paper by J. Millen and V. Shmatikov [205], many results (*e.g.* [114, 41]) have been obtained within this framework.

Here, we consider only symmetric/asymmetric encryptions, and pairing. We show that any deducibility constraint system can be transformed in (possibly several) much simpler deducibility constraint systems that are called solved forms, preserving all solutions of the original system, and not only its satisfiability. In other words, the deducibility constraint system represents in a symbolic way all the possible sequences of messages that are produced, following the protocol rules, whatever are the intruder's actions. This set of symbolic traces is infinite in general. Solved forms are a simple (and finite) representation of such traces. The procedure preserves all solutions. Hence, we can represent for instance, all attacks on the secrecy and not only decide if there exists one. Moreover, presenting the decision procedure using a small set of simplification rules yields more flexibility for further extensions and modifications.

4.1 Intruder Deduction problem

4.1.1 Preliminaries

An *inference rule* is a rule of the form $\dfrac{u_1 \;\dots\; u_n}{u_0}$ where $u_0, u_1, \dots, u_n$ are terms (with variables). An *inference system* is a set of inference rules.

Example 4.1 *The following inference system $\mathcal{I}_{\mathsf{DY}}$ represents the deduction capabilities of an attacker. We consider the signature $\mathcal{F} = \{\mathsf{senc}, \mathsf{aenc}, \langle \_, \_ \rangle, \mathsf{sk}\}$ and the underlying rewriting system $\mathcal{R}$ is empty. There are several possible ways of defining the intruder capabilities, we choose here the "implicit destructors" formulation, in which the destructors do not appear. This leads to an inference system that is slightly different from the one proposed in Section 3.1.3. For sake of simplicity, we make a confusion between the identity of an agent and his public key.*

$$\frac{x \quad y}{\langle x, y\rangle}\ \mathsf{P} \qquad\qquad \frac{x \quad y}{\mathsf{aenc}(x,y)}\ \mathsf{PKE} \qquad\qquad \frac{x \quad y}{\mathsf{senc}(x,y)}\ \mathsf{SE}$$

$$\frac{\langle x, y\rangle}{x}\ \mathsf{Left} \qquad \frac{\langle x, y\rangle}{y}\ \mathsf{Right} \qquad\qquad \frac{\mathsf{aenc}(x,y) \quad \mathsf{sk}(y)}{x}\ \mathsf{PKD} \qquad\qquad \frac{\mathsf{senc}(x,y) \quad y}{x}\ \mathsf{SD}$$

The rules P*,* SE*, and* PKE *are* composition rules *whereas the rules* Left*,* Right*,* SD*, and* PKD *are* decomposition rules.

Definition 4.1 (proof) *Let $\mathcal{I}$ be an inference system. A proof Π of $T \vdash u$ in $\mathcal{I}$ is a tree such that:*

- *every leaf of Π is labelled with a term v such that $v \in T$,*
- *for every node labelled with v_0 having n sons labelled with $v_1, \dots, v_n$, there is an instance of an inference rule with conclusion v_0 and hypotheses $v_1, \dots, v_n$. We say that Π ends with this instance if the node is the root of Π,*
- *the root is labelled with u.*

We denote by $\mathsf{Hyp}(\Pi)$ the set of labels of the leaves of a proof Π and $\mathsf{Conc}(\Pi)$ is the label of the root of Π. $\mathsf{Steps}(\Pi)$ is the set of labels of all nodes of Π. The *size* of a proof Π is the number of nodes in it. A proof Π of $T \vdash u$ is *minimal* if it does not exist any proof Π' of $T \vdash u$ having a size strictly smaller than the size of Π.

Example 4.2 *Let $\phi = \mathsf{new}\ a, b, s.\ \{^{\langle \mathsf{senc}(s,\langle a,b\rangle),a\rangle}/_{x_1},\ ^{\mathsf{senc}(b,a)}/_{x_2}\}$. We may ask whether s is deducible from ϕ,* i.e. *does there exist a proof of $\langle \mathsf{senc}(s,\langle a,b\rangle),a\rangle,\ \mathsf{senc}(b,a) \vdash s$. Such a proof is given below:*

$$\dfrac{\dfrac{\langle \mathsf{senc}(s,\langle a,b\rangle),a\rangle}{\mathsf{senc}(s,\langle a,b\rangle)} \qquad \dfrac{\dfrac{\langle \mathsf{senc}(s,\langle a,b\rangle),a\rangle}{a} \qquad \dfrac{\mathsf{senc}(b,a) \qquad \dfrac{\langle \mathsf{senc}(s,\langle a,b\rangle),a\rangle}{a}}{b}}{\langle a,b\rangle}}{s}$$

The problem whether an intruder can gain certain information s from a set of knowledge T, *i.e.* whether there is a proof of $T \vdash s$ is called the *intruder deduction problem.*

Intruder deduction problem (for a fixed inference system $\mathcal{I}$)

INPUT: a finite set of terms T, and a term s (the secret).

OUTPUT: Does there exist a proof of $T \vdash s$?

This definition is in-line with the concept of deduction introduced in Section 3.1.3. Here, we do not explicitly rely on the concept of frame. Note that for deduction, the ordering in which the messages have been sent is not relevant. Moreover, restriction on names are not necessary. It is assumed that each name is restricted.

4.1.2 Decidability via Locality

To show that the intruder deduction problem is decidable (in PTIME) for an inference system $\mathcal{I}$, we use the notion of *locality* introduced by D. McAllester [201].

Definition 4.2 (locality) *Let $\mathcal{I}$ be an inference system. The system $\mathcal{I}$ is* local *if whenever $T \vdash u$ in $\mathcal{I}$, there exists a proof Π of $T \vdash u$ such that $\mathsf{Steps}(\Pi) \subseteq st(T \cup \{u\})$.*

Given an inference system $\mathcal{I}$, to establish that the intruder deduction problem is decidable, it is actually sufficient to prove that:

1. *a locality result* for the inference system $\mathcal{I}$: checking the existence of a proof of $T \vdash u$ amounts to checking the existence of a local proof that only contains subterms of u and T (there is a polynomial number of subterms),

2. *a one-step-deducibility result* to ensure that we can test (in PTIME) whether a term is deducible in one step from a set of terms by using an instance of one of the inference rules. This result trivially holds for the inference system presented in Example 4.1.

Then, the existence of a local proof of $T \vdash u$ can be checked in polynomial time by saturation of T with terms deducible in one-step. Thanks to locality, the number of iteration to obtain a saturated set is bounded by the number of terms that can be involved in a local proof. This yields a PTIME algorithm.

Lemma 4.1 (locality) *Let T be a set of terms and u be a term. A minimal proof Π of $T \vdash u$ only contains terms in $st(T \cup \{u\})$,* i.e. $\mathsf{Steps}(\Pi) \subseteq st(T \cup \{u\})$. *Moreover, if Π is reduced to a leaf or ends with a decomposition rule, then we have that $\mathsf{Steps}(\Pi) \subseteq st(T)$.*

Proof : Let Π be a minimal proof of $T \vdash u$. We prove the result by induction on the size of the proof Π.

Base case: In such a case, the proof Π is reduced to a leaf and we easily conclude.

Induction step: We have that:

$$\Pi = \left\{ \begin{array}{c} \begin{array}{ccc} \Pi_1 & & \Pi_n \\ u_1 & \cdots & u_n \end{array} \\ \hline u \end{array} \right. \mathsf{R}$$

We distinguish several cases depending on the last inference rule of Π.

- If R is a composition rule, then $u_1, \ldots, u_n$ are subterms of u and we easily conclude by relying on our induction hypothesis.

- If R is a projection rule (say proj_1), then $u_1 = \langle u, v \rangle$ for some v. In such a case, by minimality of Π, we know that Π_1 does not end with a composition rule. Hence, by relying on our induction hypothesis, we have that $\mathsf{Steps}(\Pi_1) \subseteq st(T)$, and thus $u_1 \in st(T)$. Moreover, we have that $u \in st(u_1)$, and thus $u \in st(T)$. This allows us to conclude that $\mathsf{Steps}(\Pi) \subseteq st(T)$.

The cases where Π ends with a decryption rule (symmetric and asymmetric) can be done in a similar way. □

Proposition 4.1 *The intruder deduction problem is decidable in PTIME for $\mathcal{I}_{\mathsf{DY}}$. Actually, this problem is PTIME complete.*

The PTIME-hardness can be proved by a reduction from HORNSAT.

The concept of locality has been used to establish decidability of several inference systems. For instance, we may want to model digital signature, exclusive or operator, commutative encryption, ...

4.2 Deducibility constraints

Assume processes without replication. Then the transition system is finite in depth but might be infinitely branching, as we saw in Example 3.14. The idea then is to represent in a simple symbolic way the set of terms that satisfy the required conditions. This is what we formalise now.

Definition 4.3 *A* Deducibility constraint system *is either $\bot$ or a conjunction of deducibility constraints of the form:*

$$T_1 \overset{?}{\vdash} u_1 \wedge \ldots \wedge T_n \overset{?}{\vdash} u_n$$

in which $T_1, \ldots, T_n$ are finite sets of terms, $u_1, \ldots, u_n$ are terms. Moreover, we assume that the constraints can be ordered in such a way that:

- *monotonicity: $\emptyset \neq T_1 \subseteq T_2 \cdots \subseteq T_n$*
- *origination: for every i, we have that $fv(T_i) \subseteq fv(u_1, \ldots, u_{i-1})$*

Intuitively, the sets T_i correspond to messages that have been sent on the network, while $u_1, \ldots, u_n$ are the messages that are expected by the processes, hence have to be constructed by the environment. The first condition, called *monotonicity* reflects the fact that the set of messages that have been sent on the network can only increase. In other words, the ordering on the atomic deducibility constraints is a temporal ordering of actions. The second condition (called *origination*) reflects the properties of our processes: variables that occur in a message sent on the network must appear before in messages received from the network.

Definition 4.4 (T_x) *Let $\mathcal{C} = T_1 \overset{?}{\vdash} u_1 \ \wedge \ \ldots \ \wedge \ T_n \overset{?}{\vdash} u_n$ be a deducibility constraint system and x be a variable that occurs in $\mathcal{C}$. T_x is the minimal set (w.r.t. inclusion) among the sets $T_1, \ldots, T_n$ such that $T \overset{?}{\vdash} u \in \mathcal{C}$ and $x \in fv(u)$.*

Thanks to the monotonicity and the origination properties, for any $x \in fv(\mathcal{C})$, the set T_x exists and is uniquely defined.

Such constraint systems may be enriched with equations/disequations between terms or other constraints, that correspond to the conditions in the process calculus. We consider (for now) only these simple constraints.

Definition 4.5 (solution) *Let $\mathcal{I}$ be an inference system. A substitution σ is a* solution *of a deducibility constraint system $\mathcal{C} = T_1 \overset{?}{\vdash} u_1 \wedge \ldots \wedge T_n \overset{?}{\vdash} u_n$ if there exists a proof of $T_i\sigma \vdash u_i\sigma$ in $\mathcal{I}$ for every $i \in \{1, \ldots, n\}$.*

Example 4.3 *Consider the constraints corresponding to one of the possible Needham-Schroeder symbolic trace. We give explicitly the free names to the attacker and assume that all names that are not explicitly given are (supposedly) secret:*

$$\mathcal{C} \hat{=} \left\{ \begin{array}{rcl} a,\ b,\ d,\ \mathsf{sk}(d),\ \mathsf{aenc}(\langle a, N_a\rangle, d) & \overset{?}{\vdash} & \mathsf{aenc}(\langle a, x\rangle, b) \\ a,\ b,\ d,\ \mathsf{sk}(d),\ \mathsf{aenc}(\langle a, N_a\rangle, d),\ \mathsf{aenc}(\langle x, N_b\rangle, a) & \overset{?}{\vdash} & \mathsf{aenc}(\langle N_a, y\rangle, a) \end{array} \right.$$

The failure of the secrecy of N_b (for this scenario) is given by the additional constraint:

$$a,\ b,\ d, \mathsf{sk}(d), \mathsf{aenc}(\langle a, N_a\rangle, d),\ \mathsf{aenc}(\langle x, N_b\rangle, a),\ \mathsf{aenc}(y, d) \overset{?}{\vdash} N_b$$

A solution of $\mathcal{C}$ in $\mathcal{I}_{\mathsf{DY}}$ is $\sigma = \{x \mapsto N_a, y \mapsto N_b\}$.

4.3 Decision Procedure

We describe here a non-deterministic simplification procedure. It can be simplified in many respects, but we will see that the problem of deciding whether a constraint system has at least one solution is NP-complete anyway (for the $\mathcal{I}_{\mathsf{DY}}$ inference system given in Example 4.1). Many parts of this section, including the set of simplification rules, are borrowed from [113].

4.3.1 Simplification Rules

We prove that any deducibility constraint system can be transformed into simpler ones, called *solved*. Such simplified constraints are then used to decide the security properties.

R_1	$\mathcal{C} \wedge T \overset{?}{\vdash} u \rightsquigarrow \mathcal{C}$	if $T \cup \{x \mid (T' \overset{?}{\vdash} x) \in \mathcal{C}, T' \subsetneq T\} \vdash u$
R_2	$C \wedge T \overset{?}{\vdash} u \rightsquigarrow_\sigma \mathcal{C}\sigma \wedge T\sigma \overset{?}{\vdash} u\sigma$	if $t \in st(T), \sigma = \mathsf{mgu}(t,u), t \neq u$ t, u not variables
R_3	$\mathcal{C} \wedge T \overset{?}{\vdash} u \rightsquigarrow_\sigma \mathcal{C}\sigma \wedge T\sigma \overset{?}{\vdash} u\sigma$	if $t_1, t_2 \in st(T), \sigma = \mathsf{mgu}(t_1, t_2)$, and $t_1 \neq t_2$
R_4	$\mathcal{C} \wedge T \overset{?}{\vdash} u \rightsquigarrow \perp$	if $fv(T \cup \{u\}) = \emptyset$ and $T \not\vdash u$
R_f	$\mathcal{C} \wedge T \overset{?}{\vdash} \mathsf{f}(u,v) \rightsquigarrow \mathcal{C} \wedge T \overset{?}{\vdash} u \wedge T \overset{?}{\vdash} v$	for $\mathsf{f} \in \{\langle\ ,\ \rangle, \mathsf{senc}, \mathsf{aenc}\}$

Figure 4.1: Simplification rules.

All the rules are indexed by a substitution (when there is no index then the identity substitution is assumed). We write $\mathcal{C} \rightsquigarrow^*_\sigma \mathcal{C}'$ if there are constraint systems $\mathcal{C}_1, \ldots, \mathcal{C}_n$ such that $\mathcal{C} \rightsquigarrow_{\sigma_0} \mathcal{C}_1 \rightsquigarrow_{\sigma_1} \ldots \rightsquigarrow_{\sigma_n} \mathcal{C}'$ and $\sigma = \sigma_0\sigma_1 \ldots \sigma_n$. We denote by $\sigma = \mathsf{mgu}(u, v)$ a most general unifier of u and v, such that $fv(v\sigma, u\sigma) \subseteq fv(v, u)$.

A constraint system is called *solved* if it is different from $\perp$ and if each of its constraints is of the form $T \overset{?}{\vdash} x$, where x is a variable. Note that the empty constraint system is solved. Solved constraint systems are particularly simple since they always have a solution. Indeed, let T_1 be the smallest (w.r.t. inclusion) left-hand side of a constraint. From the definition of a constraint system we have that $T_1 \neq \emptyset$ and has no variable. Then the substitution τ defined by $x\tau = t_1$ where $t_1 \in T_1$ for every variable x, is a solution since $T \vdash x\theta$ for any constraint $T \overset{?}{\vdash} x$ of the solved constraint system.

Given a constraint system $\mathcal{C}$, we say that T_i is a *minimal unsolved left-hand side* of $\mathcal{C}$ if T_i is a left-hand side of $\mathcal{C}$ and for all $T \overset{?}{\vdash} u \in \mathcal{C}$ such that $T \subsetneq T_i$, we have that u is a variable.

Lemma 4.2 *The simplification rules transform a deducibility constraint system into a deducibility constraint system.*

Theorem 4.1 *Let $\mathcal{C}$ be an unsolved constraint system.*

1. *(Termination) There is no infinite chain $\mathcal{C} \rightsquigarrow_{\sigma_1} \mathcal{C}_1 \ldots \rightsquigarrow_{\sigma_n} \mathcal{C}_n$.*

2. *(Correctness) If $\mathcal{C} \rightsquigarrow^*_\sigma \mathcal{C}'$ for some constraint system $\mathcal{C}'$ and some substitution σ and if θ is a solution of $\mathcal{C}'$ then $\sigma\theta$ is a solution of $\mathcal{C}$.*

3. *(Completeness) If θ is a solution of $\mathcal{C}$, then there exist a solved constraint system $\mathcal{C}'$ and substitutions σ, θ' such that $\theta = \sigma\theta'$, $\mathcal{C} \rightsquigarrow^*_\sigma \mathcal{C}'$ and θ' is a solution of $\mathcal{C}'$.*

Termination and correctness are quite easy to show. For termination, it is easy to see that the number of variables is non-increasing. Furthermore, this number strictly decreases by the rules R_2 and R_3. Any other rule strictly reduces the total size of the right hand sides of the constraint (here, the "size" is the number of symbols in the term). Completeness is more involved and its proof is detailed in Section 4.3.2. Getting a polynomial bound on the length of simplification sequences requires to consider a particular strategy.