Typing messages for free in security protocols

Rémy Chrétien, Véronique Cortier, Antoine Dallon, Stéphanie Delaune

Typing messages for free in security protocols

RÉMY CHRÉTIEN, Independent researcher, France
VÉRONIQUE CORTIER, LORIA, CNRS, France
ANTOINE DALLON, LSV, CNRS & ENS Paris-Saclay, Université Paris-Saclay, France
STÉPHANIE DELAUNE, Univ Rennes, CNRS, IRISA, France

ACM Reference Format:
Rémy Chrétien, Véronique Cortier, Antoine Dallon, and Stéphanie Delaune. 2019. Typing messages for free in security protocols. *ACM Trans. Comput. Logic* 1, 1, Article 1 (January 2019), 52 pages. https://doi.org/10.1145/3343507

Abstract. Security properties of cryptographic protocols are typically expressed as reachability or equivalence properties. Secrecy and authentication are examples of reachability properties while privacy properties such as untraceability, vote secrecy, or anonymity are generally expressed as behavioral equivalence in a process algebra that models security protocols.

Our main contribution is to reduce the search space for attacks for reachability as well as equivalence properties. Specifically, we show that if there is an attack then there is one that is well-typed. Our result holds for a large class of typing systems, a family of equational theories that encompasses all standard primitives, and protocols without else branches. For many standard protocols, we deduce that it is sufficient to look for attacks that follow the format of the messages expected in an honest execution, therefore considerably reducing the search space.

CCS classification: Security and privacy/Formal methods and theory of security/Logic and verification

Keywords: security protocols, symbolic model, verification, trace equivalence

1 INTRODUCTION

Formal methods have been very successful for the analysis of security protocols and many decision procedures and tools (*e.g.* [27, 39, 40]) have been proposed. Two main families of security properties are typically considered: trace or accessibility properties, as well as equivalence properties. The former are used to express the most standard properties such as secrecy and authentication: for any execution of the protocol, an attacker should not learn the secret nor get authenticated without the server having accepted her request. The later model privacy properties such as untraceability, vote secrecy, or anonymity (*e.g.* [9, 17]). For example, the anonymity of Bob is typically expressed by the fact that an adversary should not distinguish between the situation where Bob is present and the situation where Alice is present. Formally, the behavior of a protocol can be modeled through a process algebra such as CSP or the pi calculus, enriched with terms to

Authors' addresses: Rémy Chrétien, Independent researcher, France; Véronique Cortier, LORIA, CNRS, Nancy, France, veronique.cortier@loria.fr; Antoine Dallon, LSV, CNRS & ENS Paris-Saclay, Université Paris-Saclay, Cachan, France, dallon@lsv.fr; Stéphanie Delaune, Univ Rennes, CNRS, IRISA, Rennes, France, stephanie.delaune@irisa.fr.

1529-3785/2019/1-ART1 $15.00
https://doi.org/10.1145/3343507

represent cryptographic messages. Then indistinguishability can be modeled through various behavioral equivalences. In contrast, secrecy or authentication are typically expressed by requesting that some bad event never occurs or that some event (e.g. Bob is logged) is always preceded by another one (e.g. the server granted access to Bob). Then checking for privacy amounts into checking for trace equivalence between processes, while checking for secrecy amounts into checking that a process never reaches a certain state. Both properties are undecidable in general [33]. Many results have been developed in the context of reachability properties. For an unbounded number of sessions, several decidable classes have been identified (e.g. bounding the size of messages [33] or the number of variables [26]). Tools like ProVerif [12] or Tamarin [42] do not try to *decide* security: instead they propose sound procedures, incomplete but that work well in practice. Tamarin also supports user guidance through lemmas and direct interactions. The case of a bounded number of sessions is known to be (co)-NP-complete [41] and several tools aim at developing efficient decision procedures in practice, like AVISPA [8], or Scyther [31] (that can also handle an unbounded number of sessions). Results for equivalence properties are more rare. Even in the case of a bounded number of sessions, there are few decidability results and the associated decision procedures are complex [11, 20, 43].

Our contribution. We consider here a different approach. Instead of trying directly to decide security, we develop a simplification result. Our main contribution reduces the search space for attacks: if there is an attack, then there exists a well-typed attack. More formally, we show that if there is a witness (*i.e.* a trace) that $P \not\approx Q$ then there exists a witness which is well-typed w.r.t. P or Q. Similarly for reachability, for any trace of P, we show that there is a well-typed trace of P that follows the same sequence of inputs and outputs, on the same channels, hence preserving secrecy or authentication violations. We can consider arbitrary processes and a family of equational theories that can express most standard primitives: randomized and deterministic asymmetric and symmetric encryptions, signatures, hash, MACs, and even some form of threshold encryption (1 out of n keys or n out of n). Our class of equational theories mimics the standard equations used e.g. for encryption and signatures, so that any variant can be considered. It still has various restrictions (e.g. at most one non-linear variable) that excludes several theories of interest such as exclusive or. We provide various counter-examples to our simplification result when the equational theory deviates from our assumptions. Our results hold for any typing system provided that any two unifiable encrypted subterms of P (or Q) are of the same type. It is then up to the user to adjust the typing system such that this hypothesis holds for the protocols under consideration. The finer the typing system is, the more our typing result restricts the attack search. One way to enforce our assumption is to consider the class of tagged protocols introduced by Blanchet and Podelski [15]. An easy way to achieve this in practice is by labeling encryption and is actually a good protocol design principle [2, 34]. Our small attack result holds for a notion of equivalence called trace equivalence with static inclusion. Intuitively, P and Q are in equivalence if any trace of P can be matched with a trace of Q such that any equality in P holds in Q (but disequalities in P are not considered), and conversely. Trace equivalence with static inclusion coincides with trace equivalence with static equivalence, as well as observational equivalence [21] for *determinate* processes, that is processes such that a sequence of actions may only yield one sequence of message, up to static equivalence.

We extend here a preliminary result presented at Concur'14 [23]. Compared to [23], we considerably enrich the class of cryptographic primitives since [23] considers (deterministic) symmetric encryption only. Moreover, we provide a small attack property for equivalence as well as reachability properties while [23] focuses on equivalence. The reachability case is somehow an intermediary step of the equivalence case. Therefore the reachability case was in some sense contained in [23]

but not formally stated as an independent result. The simplification result of [23], that we extend, has already been used in several contexts.

- First, in [23] itself, the small attack property is shown to imply decidability for an unbounded number of sessions but for simple protocols (a syntactic subclass of determinate protocols) with no fresh nonces nor fresh keys. Such a decidability result should probably extend to our novel class of primitives but we chose to focus here on the small attack property.
- [24] establishes the first decidability result for equivalence of protocols *with* fresh nonces and keys. It uses as a preliminary that only well-typed traces need to be considered.
- SAT-Equiv [28] is a new and efficient tool for deciding trace equivalence for a bounded number of sessions. Thanks to the small attack property, trace equivalence is reduced to finite model-checking and SAT-Equiv adapts standard model-checking techniques, namely graph planning.

Of course, due to the limitations of [23], these three results hold for protocols with symmetric key only. Our extension of the small attack property to a general class of primitives opens the way to new decidability results or more efficient procedures for more general primitives.

Related work. The analysis of security protocols using formal methods was very successful in discovering new attacks and providing better guarantees. It led to a large number of decision procedures and tools (*e.g.* [27, 39, 40]) have been proposed. Most of these results focus on reachability properties such as confidentiality or authentication. Much fewer results exist for behavioral equivalences. Based on a procedure proposed by Baudet [11], a first decidability result has been proposed for simple processes without else branches, and for equational theories that capture most standard primitives [21]. Then Tiu and Dawson [43] have designed and implemented a procedure for open bisimulation, a notion of equivalence stronger than the standard notion of trace equivalence. Cheval *et al* [20] have proposed and implemented a procedure for processes with else branches and standard primitives. The tool AkisS [19] is also dedicated to trace equivalence but is not guaranteed to terminate. All these results focus on a bounded number of sessions. The tools ProVerif [14] and Tamarin [42] can handle equivalence properties for an unbounded number of sessions. They actually reason on a stronger notion of equivalence (which may turn to be too strong in practice) and are not guaranteed to terminate. For an unbounded number of sessions, a few results have been established [22–24].

Our goal here is not to decide equivalence or reachability properties but to restrict the search space for attacks. As already discussed, our result extends the preliminary result of [23] to a wide class of cryptographic primitives and to reachability properties. In terms of proof techniques, we completely reshaped the proof. [23] relies on the fact that reachability is decidable for a bounded number of sessions, builds a decision algorithm for equivalence, for a bounded number of sessions, and then use this algorithm to show that, from any witness of non equivalence, it is possible to construct a well-typed witness. Our proof here directly builds a well-typed witness, without the need of decision algorithms. Moreover, the fact that [23] considers symmetric key only simplifies the proof as an attacker never needs to construct when it tries to learn new information. This is no longer true for example with asymmetric encryption or hashes where the attacker may need to encrypt and hash to compare values.

Our proof technique is inspired from the approach developed by Arapinis *et al* [7] for bounding the size of messages of an attack for the reachability case. Specifically, they show for some class of tagged protocols, that whenever there is an attack, there is a well-typed attack (for a particular typing system). We extend their approach to trace equivalence for slightly more general typing systems, and more general cryptographic primitives. Indeed, we consider a class of primitives that

encompasses all the standard ones while [7] considers a fixed set of primitives (e.g. without randomized encryption). Regarding type systems, we can for example allow type systems that are not structure preserving. One of the first small model properties has been established by Lowe [38]. It shows that it is sufficient to consider a finite number of roles, but assumes that messages are strongly typed: agents expects messages that follows a given (fixed) format and may never accept e.g. an arbitrary message instead of a nonce. Heather *et al* [35] provide a result for limiting attacks to well-typed ones, assuming a strong labeling scheme and no blind copies. Their seminal work only includes very basic types such as nonces, agents, or keys and does not consider composed keys. Ramanujam and Suresh [40] also show some kind of typing result, assuming an even stronger labeling scheme, with fresh nonces, and use it to establish a decidability result for protocols with nonces, with no blind copies and the standard primitives. Again for reachability, Mödersheim *et al* [4, 36] establish a typing result similar to our result, for a flexible class of primitives (but incomparable to ours). The type-compliance notion is more restrictive since even pairs of terms of a protocol should be non unifiable (or have the same type).

2 MODEL

2.1 Term algebra

Private data are represented through an infinite set of *names* $\mathcal{N}$. Names can model e.g. long-term and short-term keys, or nonces. We consider an infinite set Σ_0^- of constants to represent public data, or any data known by the attacker, such as agent names or attacker's nonces or keys. We also consider two additional infinite sets of constants $\Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$, and $\Sigma_{\mathsf{fresh}}^{\mathsf{bitstring}}$ which we will rely on for our technical development. We write $\Sigma_{\mathsf{fresh}} = \Sigma_{\mathsf{fresh}}^{\mathsf{atom}} \uplus \Sigma_{\mathsf{fresh}}^{\mathsf{bitstring}}$, and $\Sigma_0^+ = \Sigma_0^- \uplus \Sigma_{\mathsf{fresh}}$. Lastly, we consider two sets of *variables* $\mathcal{X}$ and $\mathcal{W}$. Variables in $\mathcal{X}$ typically model arbitrary data expected by the protocol, while variables in $\mathcal{W}$ are used to store messages learnt by the attacker. All these sets are assumed to be pairwise disjoint. A *data* is either a constant, a variable, or a name.

We model messages exchanged on the network and computations of the attacker by *terms*. A *signature* Σ is a set of function symbols with their arity. We distinguish between *constructor* symbols, like encryption, in Σ_{c} and *destructor* symbols, like decryption, in Σ_{d}, i.e. $\Sigma = \Sigma_{\mathrm{c}} \uplus \Sigma_{\mathrm{d}}$. Given a signature $\mathcal{F}$, the set of terms built from $\mathcal{F}$ and a set of data D is denoted $\mathcal{T}(\mathcal{F}, D)$. *Constructor terms* on D are terms in $\mathcal{T}(\Sigma_{\mathrm{c}}, D)$. We use the usual terminology on terms, that we recall here. We denote $vars(u)$ the set of variables that occur in a term u. A term is *ground* if it contains no variable. The application of a substitution σ to a term u is written $u\sigma$. We denote $dom(\sigma)$ its *domain* and $img(\sigma)$ its *image*. The *positions* of a term are defined as usual. Given a term t, we denote $\mathrm{root}(t)$ the function symbol occurring at position ϵ in t, and $St(t)$ its set of *subterms*. Two terms u_1 and u_2 are *unifiable* when there exists a substitution σ such that $u_1\sigma = u_2\sigma$.

We consider two *sorts*: atom and bitstring. atom represents atomic data like nonces or keys while bitstring models arbitrary messages. Names in $\mathcal{N}$, and constants in $\Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$ have sort atom, whereas constants in $\Sigma_{\mathsf{fresh}}^{\mathsf{bitstring}}$ have sort bitstring. The set Σ_0^- contains an infinite number of constants of both sorts. The constants of sort atom are called *atomic constants*. Any constructor f comes with its sort, i.e. $\mathsf{f} : (s_1 \times \cdots \times s_n) \to s_0$ where n is the arity of f, $s_0 = \mathsf{bitstring}$, and $s_i \in \{\mathsf{atom}; \mathsf{bitstring}\}$ for $0 \leq i \leq n$. Given a constructor term $t \in \mathcal{T}(\Sigma_{\mathrm{c}}, \Sigma_0^+ \uplus \mathcal{X})$, p is an *atomic position* of t if it corresponds to a position where an atom is expected, that is,

$$p = p'.i, \quad t|_{p'} = \mathsf{f}(t_1, \ldots, t_n) \text{ for some } \mathsf{f} \in \Sigma_{\mathrm{c}} : (s_1 \times \cdots \times s_n) \to s_0, \text{ and } s_i = \mathsf{atom}.$$

We say that t is *well-sorted* if any of its subterm is of the right sort, that is, $t|_p \in \Sigma_0^- \uplus \Sigma_{\mathsf{fresh}}^{\mathsf{atom}} \uplus \mathcal{X}$ for any atomic position p of t.

Example 2.1. Randomized asymmetric encryption, pairs, and triples are typically modelled by the following signature

$$\Sigma^{\mathrm{ex}} = \{\mathsf{raenc}, \mathsf{pub}, \mathsf{radec}, \langle\,\rangle, \langle\,\rangle^3, \mathsf{fst}, \mathsf{snd}, \mathsf{proj}_1^3, \mathsf{proj}_2^3, \mathsf{proj}_3^3\}$$

with $\Sigma_{\mathrm{c}}^{\mathrm{ex}} = \{\mathsf{raenc}, \mathsf{pub}, \langle\,\rangle, \langle\,\rangle^3\}$, and $\Sigma_{\mathrm{d}}^{\mathrm{ex}} = \{\mathsf{radec}, \mathsf{fst}, \mathsf{snd}, \mathsf{proj}_1^3, \mathsf{proj}_2^3, \mathsf{proj}_3^3\}$.

The symbols raenc (arity 3) and radec (arity 2) represent resp. randomized asymmetric encryption and decryption. The symbol pub is a key function that models the public key associated to a given private key. Pairing is modeled using $\langle\,\rangle$ of arity 2, whereas associated projection functions are denoted fst and snd (both of arity 1). We also model triples by $\langle\,\rangle^3$ of arity 3. Projection functions proj_i^3 ($i \in \{1;2;3\}$) are of arity 1 and retrieve the component x_i of the triple $\langle x_1, x_2, x_3\rangle^3$. The sort of our constructors are as follows:

$$\begin{aligned}
\mathsf{raenc}: &\quad \mathsf{bitstring} \times \mathsf{bitstring} \times \mathsf{atom} \longrightarrow \mathsf{bitstring}\\
\mathsf{pub}: &\quad \mathsf{atom} \longrightarrow \mathsf{bitstring}\\
\langle\,\rangle: &\quad \mathsf{bitstring} \times \mathsf{bitstring} \longrightarrow \mathsf{bitstring}\\
\langle\,\rangle^3: &\quad \mathsf{bitstring} \times \mathsf{bitstring} \times \mathsf{bitstring} \longrightarrow \mathsf{bitstring}
\end{aligned}$$

Let $a, b \in \Sigma_0^-$ and $sk_0, r_0 \in \mathcal{N}$ (all of sort atom), the term $u_0 = \mathsf{raenc}(\langle a, b\rangle, \mathsf{pub}(sk), r_0)$ represents the pair $\langle a, b\rangle$ encrypted by the public key $\mathsf{pub}(sk)$ using randomness r_0. The atomic positions of u_0 are $p_1 = 2.1$ and $p_2 = 3$. We chose to model randomness as an atom to reflect that randoms are typically non composed messages. However, we could consider the function symbol raenc with sort: $\mathsf{bitstring} \times \mathsf{bitstring} \times \mathsf{bitstring} \longrightarrow \mathsf{bitstring}$ as well. In such a situation, $p_1 = 2.1$ would be the only atomic position of u_0.

We consider theories where, intuitively, each symbol corresponds to a particular function that may be applied only in one particular context. For example, if asymmetric encryption is represented by $\mathsf{aenc}(m, \mathsf{pk}(k))$ then it should not be applied to other keys, such as $\mathsf{vk}(k)$. We define as a *shape*, the expected pattern of a message. Intuitively, when a term has a right shape then the application of a rewrite rule may fail only because of disequalities (e.g. attempting to decrypt with the wrong key) but not because of a mismatch between the term and the form of the rewrite rule. A shape is typically more precise than the sort of head symbol. Formally, to each constructor function symbol f, we associate a *linear* term $\mathsf{f}(u_1, \ldots, u_n) \in \mathcal{T}(\Sigma_{\mathrm{c}}, \mathcal{X})$ denoted sh_{f} which is called the *shape* of f. Shapes have to be *compatible*, that is, fixed for a given function symbol. Formally, for any $\mathsf{f}(t_1, \ldots, t_n)$ occurring in a shape, we have that $\mathsf{f}(t_1, \ldots, t_n) = sh_{\mathsf{f}}$. A term is *well-shaped* if it complies with the shapes, that is, any subterm of t, heading with a constructor symbol f is an instance of the shape of f. More formally, a constructor term $t \in \mathcal{T}(\Sigma_{\mathrm{c}}, \Sigma_0^+ \uplus \mathcal{X})$ is well-shaped if for any $t' \in St(t)$ such that $\mathsf{root}(t') = \mathsf{f}$, we have that $t' = sh_{\mathsf{f}}\sigma$ for some substitution σ. In particular, this notion of compatibility will ensure the existence of well-shaped terms. Given $D \subseteq \Sigma_0^+ \uplus \mathcal{X}$, we denote $\mathcal{T}_0(\Sigma_{\mathrm{c}}, D)$ the subset of ground constructor terms on D that are well-shaped and well-sorted. Given a set Σ_0 of constants (typically Σ_0^- or Σ_0^+), Σ_0-messages are terms in $\mathcal{T}_0(\Sigma_{\mathrm{c}}, \mathcal{N} \uplus \Sigma_0)$.

Example 2.2. Continuing Example 2.1, the shapes associated to each constructor are as follows:

$$\begin{aligned}
\mathsf{raenc}: &\quad sh_{\mathsf{raenc}} = \mathsf{raenc}(x_1, \mathsf{pub}(x_2), x_3)\\
\mathsf{pub}: &\quad sh_{\mathsf{pub}} = \mathsf{pub}(x_2)\\
\langle\,\rangle: &\quad sh_{\langle\,\rangle} = \langle x_1, x_2\rangle\\
\langle\,\rangle^3: &\quad sh_{\langle\,\rangle^3} = \langle x_1, x_2, x_3\rangle^3
\end{aligned}$$

The term $u_0 = \mathsf{raenc}(\langle a, b\rangle, \mathsf{pub}(sk), r_0)$ is a constructor term. Actually this is a Σ_0^--message. We may note that it is well-sorted since sk and r_0 are of sort atom, and well-shaped. The constructor term $\mathsf{raenc}(a, sk, r_0)$ is well-sorted but it is *not* well-shaped. Note that we will require protocols to only process messages, which enforces that terms are well-sorted and well-shaped.

The properties of cryptographic primitives are represented through a set $\mathcal{R}$ of rewriting rules. We consider rewriting rules that apply a destructor on top of constructor terms that are linear, well-sorted, and well-shaped. We strictly control the non-linearity of our rules, and we assume the standard subterm property. More formally, for each destructor symbol $\mathsf{des} \in \Sigma_{\mathrm{d}}$, there is exactly one rule of the form $\ell_{\mathsf{des}} \to r_{\mathsf{des}}$ such that:

(1) $\ell_{\mathsf{des}} = \mathsf{des}(t_1, \ldots, t_n)$ where each t_i is either a variable, or equal to $sh_{\mathsf{root}(t_i)}$ up to a bijective renaming of variables;
(2) $r_{\mathsf{des}} \in \mathcal{T}_0(\Sigma_{\mathrm{c}}, \emptyset) \cup St(t_1)$. In case r_{des} is not a fixed constructor term, for simplicity and readability in our technical developments, we assume that r_{des} occurs in t_1. Of course, our results easily extend to the case where the arguments of a destructor symbol are written in a different order;
(3) either ℓ_{des} is a linear term, or there is a unique variable x with several occurrences in ℓ_{des} and such that x occurs exactly once at an atomic position in t_1.

Moreover, we assume the existence of at least one non linear rule in our set $\mathcal{R}$ of rewriting rules.

Given a set $\mathcal{R}$ of rewriting rules, a term u can be rewritten in v using $\mathcal{R}$ if there is a position p in u, and a rewriting rule $\mathrm{g}(t_1, ..., t_n) \to t$ in $\mathcal{R}$ such that $u|_p = \mathrm{g}(t_1, \ldots, t_n)\theta$ for some substitution θ, and $v = u[t\theta]_p$, i.e. u in which the subterm at position p has been replaced by $t\theta$. Moreover, we assume that $t_1\theta, \ldots, t_n\theta$ as well as $t\theta$ are Σ_0^+-messages, in particular they do not contain destructor symbols. We consider sets of rewriting rules that yield a convergent rewriting system. As usual, we denote $\to^*$ the reflexive-transitive closure of $\to$, and $u{\downarrow}$ the normal form of a term u (it is well defined as our rewriting system is convergent by unicity of the rule associated to each destructor).

Example 2.3. The properties of the primitives given in Example 2.1 are reflected through the following rewriting rules.

$$\begin{array}{rcl@{\qquad}rcll} \mathsf{fst}(\langle x, y\rangle) & \to & x & \mathsf{radec}(\mathsf{raenc}(x, \mathsf{pub}(y), z), y) & \to & x & \\ \mathsf{snd}(\langle x, y\rangle) & \to & y & \mathsf{proj}_i^3(\langle x_1, x_2, x_3\rangle_3) & \to & x_i & \text{with } i \in \{1; 2; 3\} \end{array}$$

They satisfy all the requirements stated above.

Our class of rewriting rules is flexible enough to represent most of the standard primitives as illustrated in the following example. However, we cannot model for instance a decryption algorithm that never fails and always returns a bitstring (*e.g.* $\mathsf{sdec}(m, k)$). Indeed, such a term is not a message and will not be accepted as input or output of a protocol. We cannot model either any primitive that includes an associative and commutative operator, such as the exclusive or, or the modular exponentiation.

Example 2.4. We may consider symmetric encryption (randomized or not) using the signature $\Sigma^{\mathsf{senc}} = \{\mathsf{senc}, \mathsf{rsenc}, \mathsf{sdec}, \mathsf{rsdec}\}$ and the rewriting rules:

$$\begin{array}{rll} \mathsf{senc}: & \mathsf{bitstring} \times \mathsf{atom} \to \mathsf{bitstring} & sh_{\mathsf{rsenc}} = \mathsf{senc}(x_1, x_2) \\ \mathsf{rsenc}: & \mathsf{bitstring} \times \mathsf{atom} \times \mathsf{atom} \to \mathsf{bitstring} & sh_{\mathsf{rsenc}} = \mathsf{rsenc}(x_1, x_2, x_3) \end{array}$$

$$\mathsf{sdec}(\mathsf{senc}(x, y), y) \to x \quad \mathsf{rsdec}(\mathsf{rsenc}(x, y, z), y) \to x$$

Of course, we may consider non randomized asymmetric encryption as well. We can also model signature, and hash function through the signature $\Sigma^{\mathsf{sign}} = \{\mathsf{sign}, \mathsf{getmsg}, \mathsf{check}, \mathsf{vk}, \mathsf{ok}, \mathsf{hash}\}$:

$$\begin{array}{rll} \mathsf{ok}: & \to \mathsf{bitstring} & sh_{\mathsf{ok}} = \mathsf{ok} \\ \mathsf{sign}: & \mathsf{bitstring} \times \mathsf{atom} \to \mathsf{bitstring} & sh_{\mathsf{sign}} = \mathsf{sign}(x_1, x_2) \\ \mathsf{vk}: & \mathsf{atom} \to \mathsf{bitstring} & sh_{\mathsf{vk}} = \mathsf{vk}(x_1) \\ \mathsf{hash}: & \mathsf{bitstring} \to \mathsf{bitstring} & sh_{\mathsf{hash}} = \mathsf{hash}(x_1) \end{array}$$

$$\mathsf{getmsg}(\mathsf{sign}(x, y)) \to x \qquad \mathsf{check}(\mathsf{sign}(x, y), \mathsf{vk}(y)) \to \mathsf{ok}$$

We may also represent more exotic theory like 1 out of n encryption (that is, one key among n suffices to decrypt) and n out of n encryption (that is, the n shares of the key are needed to decrypt) through the signature $\Sigma^{\mathsf{shamir}} = \{\mathsf{k}_1, \mathsf{k}_2, \mathsf{reveal}, \mathsf{get}_1, \mathsf{get}_2, \mathsf{onekey}, \mathsf{allkeys}\}$:

$$\begin{array}{rll} \mathsf{k}_i : & \mathsf{atom} \to \mathsf{bitstring} & sh_{\mathsf{k}_i} = \mathsf{k}_i(x_1); \\ \mathsf{onekey} : & \mathsf{bitstring} \times \mathsf{atom} \times \mathsf{atom} & sh_{\mathsf{onekey}} = \mathsf{onekey}(x_1, x_2, x_3); \\ \mathsf{allkeys} : & \mathsf{bitstring} \times \mathsf{atom} & sh_{\mathsf{allkeys}} = \mathsf{allkeys}(x_1, x_2); \end{array}$$

$$\mathsf{get}_i(\mathsf{onekey}(x, y_1, y_2), y_i) \to x \text{ with } i \in \{1, 2\} \qquad \mathsf{reveal}(\mathsf{allkeys}(x, y), \mathsf{k}_1(y), \mathsf{k}_2(y)) \to x$$

We can also model tuples of various size in a similar fashion than triples in Example 2.1. Another theory of interest is when it is possible to check whether two ciphertexts have been encrypted with the same key. This can be modeled by adding the destructor symbol samekey to Σ^{senc} and the rewrite rule

$$\mathsf{samekey}(\mathsf{senc}(x_1, y), \mathsf{senc}(x_2, y)) \to \mathsf{ok}$$

Note however that the theory for equality, often modeled by the simple rule $\mathsf{eq}(x, x) \to \mathsf{ok}$ does not fit our assumptions. First, since x is a non-linear variable, it needs to be of sort atom, which seriously limits test for equalities. But even with this restriction, the rewrite rule cannot be accepted since x appears directly under a destructor, there is no constructor associated to it.

An attacker builds his own messages by applying public function symbols to terms he already knows and that are available through variables in $\mathcal{W}$. Formally, given a set Σ_0 of constants (typically Σ_0^- or Σ_0^+), a computation done by the attacker is a Σ_0-*recipe*, *i.e.* a term in $\mathcal{T}(\Sigma, \mathcal{W} \uplus \Sigma_0)$.

2.2 Process algebra

Our process algebra is inspired from the applied pi calculus [1]. We do not consider else branches. Actually we do not have conditional. Instead, equality tests are performed through pattern-matching. We do not consider replication but our typing result easily extend to processes with replication as explained in Section 6 (see also [23]). Indeed, our key result shows how to build a well-typed trace from an arbitrary one. This holds for traces obtained from finite processes as well as traces from replicated processes.

Let Ch be an infinite set of *channels*. We consider processes built using the following grammar:

$$\begin{array}{rcll} P, Q := & & 0 & \text{null process} \\ & \mid & \mathsf{in}(c, u).P & \text{input} \\ & \mid & \mathsf{out}(c, u).P & \text{output} \\ & \mid & (P \mid Q) & \text{parallel} \\ & \mid & i : P & \text{phase} \end{array}$$

where $u \in \mathcal{T}_0(\Sigma_c, \Sigma_0^- \uplus \mathcal{N} \uplus \mathcal{X})$, and $c \in Ch$.

The process 0 does nothing. The process $\mathsf{in}(c, u).P$ expects a message m of the form u on channel c and then behaves like $P\sigma$ where σ is a substitution such that $m = u\sigma$. The process $\mathsf{out}(c, u).P$ emits u on channel c, and then behaves like P. The variables that occur in u are instantiated when the evaluation takes place. The process $P \mid Q$ runs P and Q in parallel. Our calculus also introduces a phase instruction, in the spirit of [14], denoted $i : P$.

For the sake of clarity, we may omit the null process. We write $fv(P)$ for the set of *free variables* that occur in P, *i.e.* the set of variables that are not in the scope of an input. We also assume that processes are *variable distinct*, *i.e.* any variable is at most bound once. For example, in the process $\mathsf{in}(c, x).\mathsf{in}(c, x)$ the variable x is bound once, whereas in the process $\mathsf{in}(c, x) \mid \mathsf{in}(c, x)$, one occurrence of the variable x should be renamed to ensure variable distinctness.

Phases are useful to specify properties inspired from cryptographic, game-based security definitions. A typical example is the real-or-random property: after interacting with a protocol, an attacker is given the real secret or a random one and she should not make the difference. Phases are also used to state unlinkability properties [10], to model that an attacker should not be able to distinguish between Alice and Bob, even *after* having interacted with Alice. Some protocols also directly include several steps, that are best modeled with phases. This is for example the case of voting protocols [16].

Example 2.5. We consider a variant of the Needham Schroeder Lowe public key protocol [37] with randomized encryption. The protocol aims at ensuring mutual authentication through the secrecy of the nonces N_a and N_b that are exchanged during an execution. It can be described informally as follows:

$$\begin{array}{lll} 1. & A \to B: & \mathsf{raenc}(\langle A, N_a\rangle, \mathsf{pub}(B), r_1) \\ 2. & B \to A: & \mathsf{raenc}(\langle N_a, \langle N_b, B\rangle\rangle, \mathsf{pub}(A), r_2) \\ 3. & A \to B: & \mathsf{raenc}(N_b, \mathsf{pub}(B), r_3) \end{array}$$

where A and B are agents trying to authenticate each other, $\mathsf{pub}(A)$ (resp. $\mathsf{pub}(B)$) is the public key of A (resp. B), N_a and N_b (as well as r_1, r_2, and r_3) are nonces generated by A and B. This is a slight variant of the original protocol [37] proposed by J. Millen: in the second message, the identity of B is placed at the end of the message, instead of the beginning of the message. This variant is subject to a type-flaw attack (discovered by J. Millen) as we shall explain in the next section.

We model the Needham Schroeder Lowe protocol in our formalism through the process P_{NSL} that results from the composition of the process P_A representing the role of A and the process P_B representing the role of B.

$$P_{\mathsf{NSL}} = P_A \mid P_B$$

with P_A and P_B defined as follows.

$$\begin{array}{lll} P_A = & \mathsf{out}(c_A, \mathsf{raenc}(\langle \mathsf{a}, n_a\rangle, \mathsf{pub}(sk_b), r_1)). & \\ & \mathsf{in}(c_A, \mathsf{raenc}(\langle n_a, \langle x_1, \mathsf{b}\rangle\rangle, \mathsf{pub}(sk_a), x_2)). & \\ & \mathsf{out}(c_A, \mathsf{raenc}(x_1, \mathsf{pub}(sk_b), r_3)) & \end{array} \qquad \begin{array}{lll} P_B = & \mathsf{in}(c_B, \mathsf{raenc}(\langle \mathsf{a}, y_1\rangle, \mathsf{pub}(sk_b), y_2)). & \\ & \mathsf{out}(c_B, \mathsf{raenc}(\langle y_1, \langle n_b, \mathsf{b}\rangle\rangle, \mathsf{pub}(sk_a), r_2)). & \\ & \mathsf{in}(c_B, \mathsf{raenc}(n_b, \mathsf{pub}(sk_b), y_3)). & \end{array}$$

where sk_a, sk_b, n_a, n_b, r_1, r_2, and r_3 are names, whereas a and b are constants from Σ_0^-.

In order to model a richer scenario, we may want to consider in addition the process P'_B that corresponds to the role B played by agent a interacting with a dishonest agent c. Below, $r'_3, n'_b \in \mathcal{N}$ whereas $\mathsf{c}, sk_c \in \Sigma_0^-$ (so that they are implicitly given to the attacker). We write $\overline{P}_{\mathsf{NSL}} = P_B \mid P'_B$

$$\begin{array}{lll} P'_B = & \mathsf{in}(c'_B, \mathsf{raenc}(\langle \mathsf{c}, y'_1\rangle, \mathsf{pub}(sk_a), y'_2)). & \\ & \mathsf{out}(c'_B, \mathsf{raenc}(\langle y'_1, \langle n'_b, \mathsf{a}\rangle\rangle, \mathsf{pub}(sk_c), r'_2)). & \\ & \mathsf{in}(c'_B, \mathsf{raenc}(n'_b, \mathsf{pub}(sk_a), y'_3)). & \end{array}$$

2.3 Semantics.

The operational semantics of a process is defined using a relation over configurations. Configurations are parameterized by a set of constants Σ_0 (typically Σ_0^- or Σ_0^+). A Σ_0-*configuration* is a tuple $(\mathcal{P}; \phi; \sigma; i)$ with $i \in \mathbb{N}$ such that:

- $\mathcal{P}$ is a multiset of processes (not necessarily ground);
- $\phi = \{\mathsf{w}_1 \triangleright m_1, \ldots, \mathsf{w}_n \triangleright m_n\}$ is a Σ_0-*frame*, *i.e.* a substitution where $\mathsf{w}_1, \ldots, \mathsf{w}_n$ are variables in $\mathcal{W}$, and $m_1, \ldots, m_n$ are Σ_0-messages;
- σ is a substitution such that $dom(\sigma) = fv(\mathcal{P})$, and $img(\sigma)$ are Σ_0-messages.

$$\textsc{In} \quad (i : \mathsf{in}(c,u).P \cup \mathcal{P}; \phi; \sigma; i) \xrightarrow{\mathsf{in}(c,R)} (i : P \cup \mathcal{P}; \phi; \sigma \uplus \sigma_0; i) \quad \text{where } R \text{ is a } \Sigma_0\text{-recipe such that } R\phi\downarrow \text{ is a } \Sigma_0\text{-message, and } R\phi\downarrow = (u\sigma)\sigma_0 \text{ for } \sigma_0 \text{ with } dom(\sigma_0) = vars(u\sigma).$$

$$\textsc{Out} \quad (i : \mathsf{out}(c,u).P \cup \mathcal{P}; \phi; \sigma; i) \xrightarrow{\mathsf{out}(c,\mathsf{w})} (i : P \cup \mathcal{P}; \phi \cup \{\mathsf{w} \triangleright u\sigma\}; \sigma; i) \quad \text{with w a fresh variable from } \mathcal{W}, \text{ and } u\sigma \text{ is a } \Sigma_0\text{-message.}$$

$$\textsc{Move} \quad (\mathcal{P}; \phi; \sigma; i) \xrightarrow{\text{phase } i'} (\mathcal{P}; \phi; \sigma; i') \quad \text{with } i' > i.$$

$$\textsc{Phase} \quad (i' : i'' : P \cup \mathcal{P}; \phi; \sigma; i) \xrightarrow{\tau} (i'' : P \cup \mathcal{P}; \phi; \sigma; i)$$

$$\textsc{Null} \quad (i : 0 \cup \mathcal{P}; \phi; \sigma; i) \xrightarrow{\tau} (\mathcal{P}; \phi; \sigma; i)$$

$$\textsc{Par} \quad (i : (P \mid Q) \cup \mathcal{P}; \phi; \sigma; i) \xrightarrow{\tau} (i : P \cup i : Q \cup \mathcal{P}; \phi; \sigma; i)$$

Fig. 1. Semantics for processes w.r.t. Σ_0

A Σ_0-configuration $(\mathcal{P}; \phi; \sigma; i)$ such that $\sigma = \emptyset$ is said *initial*.

Intuitively, $\mathcal{P}$ represents the processes that still remain to be executed; ϕ represents the sequence of messages that have been learnt so far by the attacker, σ stores the value of the variables that have already been instantiated, and i is an integer that indicates the current phase. We often write $i : P$ instead of $(P; \emptyset; \emptyset; i)$, P instead of $0 : P$ and $P \uplus \mathcal{P}$ instead of $\{P\} \uplus \mathcal{P}$. The operational semantics of a Σ_0-configuration is induced by the relation $\xrightarrow{\alpha}$ w.r.t. Σ_0 over Σ_0-configurations defined in Figure 1.

The first rule (In) allows the attacker to send to some process a term built from publicly available terms and symbols. The second rule (Out) corresponds to the output of a term by some process: the corresponding term is added to the frame of the current configuration, which means that the attacker can now access the sent term. Note that the term is output provided that it is a message. Regarding phases (rules Move, and Phase), the attacker may move to a subsequent phase whenever he wants, while processes may move to the next phase when they are done. The two remaining rules Null and Par are quite standard and are unobservable (τ action) from the point of view of the attacker.

The relation $\xrightarrow{\alpha_1 \dots \alpha_n}$ w.r.t. Σ_0 between Σ_0-configurations (where $\alpha_1 \dots \alpha_n$ is a sequence of actions) is defined as the transitive closure of $\xrightarrow{\alpha}$ w.r.t. Σ_0. Given a sequence of observable actions tr, and two Σ_0-configurations $\mathcal{K}$ and $\mathcal{K}'$, we write $\mathcal{K} \overset{\mathsf{tr}}{\Longrightarrow} \mathcal{K}'$ w.r.t. Σ_0 when there exists a sequence $\alpha_1 \dots \alpha_n$ such that $\mathcal{K} \xrightarrow{\alpha_1 \dots \alpha_n} \mathcal{K}'$ w.r.t. Σ_0 and tr is obtained from $\alpha_1 \dots \alpha_n$ by erasing all occurrences of τ.

Definition 2.6. Given a Σ_0-configuration $\mathcal{K} = (\mathcal{P}; \phi; \sigma; i)$, we denote $\mathsf{trace}_{\Sigma_0}(\mathcal{K})$ the set of traces defined as follows:

$$\mathsf{trace}_{\Sigma_0}(\mathcal{K}) = \{(\mathsf{tr}, \phi') \mid \mathcal{K} \overset{\mathsf{tr}}{\Longrightarrow} (\mathcal{P}; \phi'; \sigma'; i') \text{ w.r.t. } \Sigma_0 \text{ for some } \Sigma_0\text{-configuration } (\mathcal{P}; \phi'; \sigma'; i')\}.$$

Given a Σ_0-configuration $\mathcal{K}$, we may note that, by definition of $\mathsf{trace}_{\Sigma_0}(\mathcal{K})$, we have that $\mathsf{tr}\phi\downarrow$, i.e. tr on which we apply the substitution ϕ followed by rewriting steps to normalize the second argument of each input/ouput action, only contains Σ_0-messages.

Example 2.7. Continuing Example 2.5, consider the initial configuration $\mathcal{K}_{\mathsf{NSL}} = (\overline{P}_{\mathsf{NSL}}; \phi_0; \emptyset; 0)$ with initial knowledge $\phi_0 = \{w_a \triangleright \mathsf{pub}(sk_a),\ w_b \triangleright \mathsf{pub}(sk_b)\}$. This models that the attacker initially knows the public keys of a and b. Note that the private key of c, sk_c, is a public constant and is thus also initially known to the attacker.

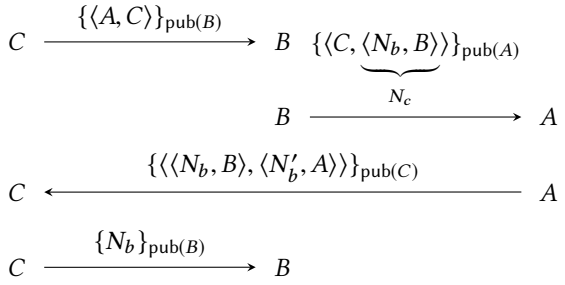

Fig. 2. Attack trace.

As mentioned in Example 2.5, this variant of the Needham Schroeder Lowe protocol is subject to a type flaw attack depicted in Figure 2. The attack relies on the fact that agent A may accept the pair $\langle N_b, B\rangle$ as a nonce N_C coming from C and agent B may confuse the identity C with a nonce. In this attack, a dishonest agent C initiates a session with agent B but writes his identity C instead of a nonce. B replies as expected by $\{C, N_b, B\}_{\mathrm{pub}(A)}$. This message is then accepted by A (playing the responder role). A thinks she is contacted by C and replies with $\{N_b, B, N'_b, A\}_{\mathrm{pub}(C)}$. Therefore the attacker learns N_b and may impersonate A w.r.t. B. This attack is reflected by the following sequence tr:

$$\mathsf{in}(c_B, \mathsf{raenc}(\langle \mathsf{a}, \mathsf{c}\rangle, \mathsf{w}_b, r_0)).\mathsf{out}(c_B, \mathsf{w}_1).\mathsf{in}(c'_B, \mathsf{w}_1).\mathsf{out}(c'_B, \mathsf{w}_2).\mathsf{in}(c_B, \mathsf{raenc}(R, \mathsf{w}_b, r'_0))$$

where $R = \mathsf{fst}(\mathsf{fst}(\mathsf{radec}(\mathsf{w}_2, sk_c)))$ and $r_0, r'_0 \in \Sigma_0^-$. This sequence of actions yields the frame ϕ defined as follows:

$$\phi = \phi_0 \uplus \{\mathsf{w}_1 \triangleright \mathsf{raenc}(\langle \mathsf{c}, \langle n_b, \mathsf{b}\rangle\rangle, \mathsf{pub}(sk_a), r_2),\ \mathsf{w}_2 \triangleright \mathsf{raenc}(\langle\langle n_b, \mathsf{b}\rangle, \langle n'_b, \mathsf{a}\rangle\rangle, \mathsf{pub}(sk_c), r'_2)\}.$$

We have that $(\mathsf{tr}, \phi) \in \mathsf{trace}(\mathcal{K}_{\mathsf{NSL}})$.

2.4 Trace equivalence

Many privacy properties such as vote-privacy or untraceability are expressed as trace equivalence [6, 32]. Intuitively, two configurations are trace equivalent if an attacker cannot tell with which of the two configurations he is interacting. We first introduce a notion of equivalence between frames.

Intuitively, an attacker can see the difference between two sequences of messages if he is able to perform some computation that succeeds in ϕ_1 and fails in ϕ_2; or if he can build a test that leads to an equality in ϕ_1 and not in ϕ_2 (or conversely).

Definition 2.8. Two Σ_0-frames ϕ_1 and ϕ_2 are in *static inclusion* w.r.t. Σ_0, written $\phi_1 \sqsubseteq_s \phi_2$ w.r.t. Σ_0, when $dom(\phi_1) = dom(\phi_2)$, and:

- for any Σ_0-recipe R, we have that $R\phi_1\downarrow$ is Σ_0-message implies that $R\phi_2\downarrow$ is Σ_0-message; and
- for any Σ_0-recipes R, R' such that $R\phi_1\downarrow, R'\phi_1\downarrow$ are Σ_0-messages, we have that: $R\phi_1\downarrow = R'\phi_1\downarrow$ implies $R\phi_2\downarrow = R'\phi_2\downarrow$.

They are in *static equivalence* w.r.t. Σ_0, written $\phi_1 \sim_s \phi_2$ w.r.t. Σ_0, if $\phi_1 \sqsubseteq_s \phi_2$ and $\phi_2 \sqsubseteq_s \phi_1$.

In the remaining of this paper, Σ_0 will be either Σ_0^- or Σ_0^+, and we sometimes omit to mention it when it is clear from the context.

Example 2.9. Continuing Example 2.7, we consider the two following Σ_0^--frames:

- $\phi_1 = \phi \uplus \{\mathsf{w}_3 \triangleright n_b\}$; and

- $\phi_2 = \phi \uplus \{w_3 \triangleright k\}$.

We have that $R\phi_1\downarrow = w_3\phi_1\downarrow$ where $R = \mathsf{fst}(\mathsf{fst}(\mathsf{radec}(w_2, sk_c)))$. This equality does not hold in ϕ_2, hence ϕ_1 and ϕ_2 are not in static equivalence.

Trace equivalence is the active counterpart of static equivalence. Two configurations are trace equivalent if, however the attacker behaves, the resulting sequences of messages observed by the attacker are in static equivalence.

Definition 2.10. A Σ_0-configuration $\mathcal{K}$ is trace included (w.r.t. Σ_0) in a Σ_0-configuration $\mathcal{K}'$, written $\mathcal{K} \sqsubseteq_t \mathcal{K}'$ w.r.t. Σ_0, if for every $(\mathsf{tr}, \phi) \in \mathsf{trace}_{\Sigma_0}(\mathcal{K})$, there exist $(\mathsf{tr}', \phi') \in \mathsf{trace}_{\Sigma_0}(\mathcal{K}')$ such that $\mathsf{tr} = \mathsf{tr}'$, and $\phi \sqsubseteq_s \phi'$ w.r.t. Σ_0. They are in trace equivalence, written $\mathcal{K} \approx_t \mathcal{K}'$ w.r.t. Σ_0, $\mathcal{K} \sqsubseteq_t \mathcal{K}'$ and $\mathcal{K}' \sqsubseteq_t \mathcal{K}$ w.r.t. Σ_0.

Note that two trace equivalent configurations are necessarily at the same phase.

This notion of trace equivalence slightly differs from the one used in [23], where the frames are required to be in static equivalence $\phi \sim_s \phi'$ instead of static inclusion $\phi \sqsubseteq_s \phi'$. Actually, these two notions of equivalence coincide for determinate protocols [19].

Definition 2.11. A configuration $\mathcal{K}$ is *determinate* if whenever $\mathcal{K} \overset{\mathsf{tr}}{\Longrightarrow} \mathcal{K}_1$ and $\mathcal{K} \overset{\mathsf{tr}}{\Longrightarrow} \mathcal{K}_2$ for some tr, $\mathcal{K}_1 = (\mathcal{P}_1; \phi_1; \sigma_1; i_1)$, and $\mathcal{K}_2 = (\mathcal{P}_2; \phi_2; \sigma_2; i_2)$. We have that $\phi_1 \sim_s \phi_2$.

Determinate protocols encompass the class of simple protocols as given e.g. in [21]. Intuitively, simple protocols are protocols such that each copy of a replicated process has its own channel reflecting the fact that due to IP addresses and sessions identifiers, an attacker can identify which process and which session he is sending messages to (or receiving messages from).

Trace equivalence can be used to express various privacy properties. A first example is anonymity: a protocol P preserves Alice's anonymity if an attacker cannot distinguish whether she is interacting with Alice or Bob.

$$P(Alice) \approx_t P(Bob)$$

where $P(Alice)$ represents a configuration where Alice is present.

Vote privacy is slightly more complicated. An attacker may know whether Alice voted or not. However, she should not be able to distinguish the scenario where Alice votes 0 and Bob votes 1, from the converse scenario where Alice votes 1 and Bob votes 0. This can be (informally) expressed through the following equivalence [32].

$$Voter(Alice, 0) \mid Voter(Bob, 1) \approx_t Voter(Alice, 1) \mid Voter(Bob, 0)$$

Stronger privacy properties such as receipt-freeness and coercion resistance can also be modeled using equivalences [32].

Another example is unlinkability in the context of IoT. For example, it should not be possible to trace Alice, even if she is wearing a biometric passport (or another tag). More precisely, even if an attacker observes (and interacts with) many sessions of Alice and other passport holders, she should not be able to distinguish between a session with Alice and a session with Bob. Assume that $P(Alice, r)$ models a session of Alice's passport (with a reader), with some fresh random r. Then unlinkability can be (informally) expressed as follows [18].

$$\begin{array}{rl} & P(A_1, r_1^1) \mid \cdots \mid P(A_1, r_1^n) \mid \cdots \mid P(A_k, r_k^1) \mid \cdots P(A_k, r_k^n) \mid 1 : P(A_1, r) \\ \approx_t & P(A_1, s_1^1) \mid \cdots \mid P(A_1, s_1^n) \mid \cdots \mid P(A_k, s_k^1) \mid \cdots P(A_k, s_k^n) \mid 1 : P(A_2, s) \end{array}$$

Finally, nonce or key secrecy can be modeled as a real-or-random game. Once the attacker is done interacting with the protocol P that establishes a secret s, then she is given either s or a fresh

random r and she should try to distinguish between these two scenario. This is often called strong secrecy.

$$P(s) \mid 1 : out(c, s) \approx_t P(s) \mid 1 : out(c, r)$$

The use of a phase is necessary to avoid false attacks. Indeed, an attacker could then try to reuse the received (real or random) secret and re-inject it in the true protocol. This real-or-random property corresponds to a stronger modeling of secrecy than standard secrecy properties modeled as reachability properties. Secrecy stated as a reachability states that the attacker is not able to compute the secret. Strong secrecy guarantees that the attacker should gain absolutely no information on the secret. For example, publishing the hash of a secret $h(s)$ preserves the secrecy of s but not its strong secrecy.

Example 2.12. Continuing Example 2.5, we consider the protocol $\overline{P}_{\mathsf{NSL}}$ that models two roles of B: one played by B responding to A, and the other one played by A responding to C. As explained above, to model the fact that the nonce n_b sent by B for A should remain secret, we require that n_b remains indistinguishable from a fresh value. Formally, we add a new phase to the process P_B. Once the attacker is done interacting with the standard protocol, then B will output either the true nonce n_b or a fresh value $k \in \mathcal{N}$. This yields the following two processes.

$$\begin{array}{lll} P_B^1 = & \mathsf{in}(c_B, \mathsf{raenc}(\langle \mathsf{a}, y_1\rangle, \mathsf{pub}(sk_b), y_2)). & \\ & \mathsf{out}(c_B, \mathsf{raenc}(\langle y_1, \langle n_b, \mathsf{b}\rangle\rangle, \mathsf{pub}(sk_a), r_2)). & \\ & \mathsf{in}(c_B, \mathsf{raenc}(n_b, \mathsf{pub}(sk_b), y_3)). & \\ & 1 : \mathsf{out}(c_B, n_b) & \end{array} \qquad \begin{array}{lll} P_B^2 = & \mathsf{in}(c_B, \mathsf{raenc}(\langle \mathsf{a}, y_1\rangle, \mathsf{pub}(sk_b), y_2)). & \\ & \mathsf{out}(c_B, \mathsf{raenc}(\langle y_1, \langle n_b, \mathsf{b}\rangle\rangle, \mathsf{pub}(sk_a), r_2)). & \\ & \mathsf{in}(c_B, \mathsf{raenc}(n_b, \mathsf{pub}(sk_b), y_3)). & \\ & 1 : \mathsf{out}(c_B, k) & \end{array}$$

The corresponding overall processes are $\overline{P}^1_{\mathsf{NSL}} = P'_B \mid P^1_B$ and $\overline{P}^2_{\mathsf{NSL}} = P'_B \mid P^2_B$, and we consider the initial frame ϕ_0 as given in Example 2.7 and the initial configurations $\mathcal{K}_1 = (\overline{P}^1_{\mathsf{NSL}}; \phi_0; \emptyset; 0)$ and $\mathcal{K}_2 = (\overline{P}^2_{\mathsf{NSL}}; \phi_0; \emptyset; 0)$. In this example, the use of a phase may not be strictly necessary w.r.t. the existence of attacks. But in principle, removing the phase could yield to false attacks.

We can show that $\mathcal{K}_1 \not\sqsubseteq_t \mathcal{K}_2$ since n_b is not strongly secret due to the attack depicted in Figure 2. This is exemplified by the trace

$$\mathsf{tr} = \mathsf{in}(c_B, R_1).\mathsf{out}(c_B, \mathsf{w}_1).\mathsf{in}(c'_B, \mathsf{w}_1).\mathsf{out}(c'_B, \mathsf{w}_2).\mathsf{in}(c_B, R_2).\mathsf{out}(c_B, \mathsf{w}_3)$$

with $R_1 = \mathsf{raenc}(\langle \mathsf{a}, \mathsf{c}\rangle, \mathsf{w}_b, r_0)$ and $R_2 = \mathsf{raenc}(\mathsf{fst}(\mathsf{fst}(\mathsf{radec}(\mathsf{w}_2, sk_c))), \mathsf{w}_b, r'_0)$, where $r_0, r'_0 \in \Sigma_0^-$. Indeed, consider now

$$\phi = \phi_0 \uplus \{\mathsf{w}_1 \triangleright \mathsf{raenc}(\langle \mathsf{c}, \langle n_b, \mathsf{b}\rangle\rangle, \mathsf{pub}(sk_a), r_2),\ \mathsf{w}_2 \triangleright \mathsf{raenc}(\langle\langle n_b, \mathsf{b}\rangle, \langle n'_b, \mathsf{a}\rangle\rangle, \mathsf{pub}(sk_c), r'_2)\}$$

and $\phi_1 = \phi_0 \uplus \{\mathsf{w}_3 \triangleright n_b\}$, $\phi_2 = \phi \uplus \{\mathsf{w}_3 \triangleright k\}$. We have $(\mathsf{tr}, \phi_1) \in \mathsf{trace}_{\Sigma_0^-}(\mathcal{K}_1)$ and $(\mathsf{tr}, \phi_2) \in \mathsf{trace}_{\Sigma_0^-}(\mathcal{K}_2)$. Now consider the equality $\mathsf{w}_3 = \mathsf{fst}(\mathsf{fst}(\mathsf{radec}(\mathsf{w}_2, sk_c)))$. It holds in ϕ_1 but not in ϕ_2 hence we have that $\mathcal{K}_1 \not\sqsubseteq_t \mathcal{K}_2$.

Consider now a variant P'_{NSL} where the second message $\{\langle N_a, \langle N_b, B\rangle\rangle\}_{\mathsf{pub}(A)}$ is no longer encoded using two nested pairs but using a triple instead, that is, $\{\langle N_a, N_b, B\rangle^3\}_{\mathsf{pub}(A)}$. This transformation yields the process

$$\begin{array}{lll} Q_B = & \mathsf{in}(c_B, \mathsf{raenc}(\langle \mathsf{a}, y_1\rangle, \mathsf{pub}(sk_b), y_2)). & \\ & \mathsf{out}(c_B, \mathsf{raenc}(\langle y_1, n_b, \mathsf{b}\rangle^3, \mathsf{pub}(sk_a), r_2)). & \\ & \mathsf{in}(c_B, \mathsf{raenc}(n_b, \mathsf{pub}(sk_b), y_3)) & \end{array}$$

instead of P_B. The use of triples rules out the type flaw attack and the resulting processes are in trace equivalence.

3 OUR TYPING RESULTS

Even when considering finite processes (i.e. processes without replication), the problem of checking trace equivalence is difficult due to several sources of unboundedness. One of them is the arbitrarily large size of messages that can be forged by an attacker. We propose here a simplification result that reduces the search space for attacks. Roughly, if there is an attack, then there is a well-typed attack, for a flexible notion of type that can be adapted depending on the desired result. We establish this result both for trace properties and equivalence properties (trace equivalence). Compared to the initial work of [23], we extend the result from a fixed, simple signature (symmetric encryption) to a large class of cryptographic primitives that encompasses all the standard ones. Moreover, [23] only applies to trace equivalence. Intuitively, proving the small attack property for trace equivalence requires to first show how to reduce the attack on a single trace. This yields a reduction result for reachability properties, such as authentication or confidentiality, which is of independent interest.

3.1 Typing system

We consider any type system that is consistent with substitution and unification.

Definition 3.1. A *typing system* is a pair $(\mathcal{T}, \delta)$ where $\mathcal{T}$ is a set of elements called *types* and δ is a function mapping terms $t \in \mathcal{T}_0(\Sigma_c, \Sigma_0^+ \cup \mathcal{N} \cup \mathcal{X})$ to types in $\mathcal{T}$ such that:

- If t is a term of type τ and σ is a well-typed substitution, that is a substitution such that x and $\sigma(x)$ have the same type, then $t\sigma$ is of type τ.
- For any unifiable terms t and t' with the same type, i.e. $\delta(t) = \delta(t')$, their most general unifier $mgu(t, t')$ is well-typed.

Consider a configuration $\mathcal{K}$ and a a typing system $(\mathcal{T}, \delta)$, an execution $\mathcal{K} \overset{\mathsf{tr}}{\Longrightarrow} (\mathcal{P}; \phi; \sigma; i)$ is *well-typed* if σ is a well-typed substitution.

Some interesting typing systems are *structure-preserving* typing systems, that is typing system that preserve the structure of terms. They are defined as follows:

Definition 3.2. A *structure-preserving typing system* is a pair $(\mathcal{T}_{\mathsf{init}}, \delta)$ where $\mathcal{T}_{\mathsf{init}}$ is a set of elements called *initial types*, and δ is a function mapping data in $\Sigma_0^+ \uplus \mathcal{N} \uplus \mathcal{X}$ to types τ generated using the following grammar:

$$\tau, \tau_1, \tau_2 = \tau_0 \mid \mathsf{f}(\tau_1, \ldots, \tau_n) \text{ with } \mathsf{f} \in \Sigma_{\mathsf{c}} \text{ and } \tau_0 \in \mathcal{T}_{\mathsf{init}}$$

Then, δ is extended to constructor terms as follows:

$$\delta(\mathsf{f}(t_1, \ldots, t_n)) = \mathsf{f}(\delta(t_1), \ldots, \delta(t_n)) \text{ with } \mathsf{f} \in \Sigma_{\mathsf{c}}.$$

The following lemma proves that structure-preserving typing systems are typing systems as defined in Definition 3.1.

Lemma 3.3. *Let $(\mathcal{T}_{\mathsf{init}}, \delta)$ be a structure-preserving typing system. Then $(\mathcal{T}(\Sigma, \mathcal{T}_{\mathsf{init}}), \delta)$ is a typing system as in Definition 3.1.*

Proof. We have to prove the two following items:

(1) If t is a term and σ is a well-typed substitution, then $\delta(t\sigma) = \delta(t)$.
(2) For any unifiable terms t and t' with the same type, i.e. $\delta(t) = \delta(t')$, their most general unifier $mgu(t, t')$ is well-typed.

We prove item (1) by induction on t. If t is a name or a constant, then $t\sigma = t$ and the result trivially holds. If t is a variable, then $\delta(t\sigma) = \delta(t)$ as σ is well-typed. Now, if $t = \mathsf{f}(t_1, \ldots, t_n)$, then $\delta(t\sigma) = \mathsf{f}(\delta(t_1\sigma), \ldots, \delta(t_n\sigma))$ as $(\mathcal{T}_{\mathsf{init}}, \delta)$ is structure-preserving. By induction hypothesis, we

have that $\delta(t_i\sigma) = \delta(t_i)$ for $i \in \{1, \ldots, n\}$, and we get that $\delta(t\sigma) = \mathsf{f}(\delta(t_1), \ldots, \delta(t_n))$. As δ is structure-preserving, we have that $\delta(t) = \mathsf{f}(\delta(t_1), \ldots, \delta(t_n))$, and this allows us to conclude.

We now prove item (2). Given a set Γ of well-typed equations, we denote $\#vars(\Gamma)$ the number of variables occurring in Γ, and $|\Gamma|$ its size, i.e. $\sum_{t=t' \in \Gamma}(|t| + |t'|)$ where $|t|$ denotes the number of symbols occurring in t. Our measure $\|\Gamma\|$ is determined by these two elements, in lexicographic order. We prove the result by induction on $\|\Gamma\|$ relying on this measure.

Base case: $\|\Gamma\| = (0, 0)$, i.e. $\Gamma = \emptyset$, and thus the result trivially holds.

Induction step: $\Gamma = \Gamma' \uplus \{t = t'\}$. We distinguish several cases:

- In case t or t' is a variable. We assume w.l.o.g. that t is a variable x. In such a case, let $\sigma = \{x \mapsto t'\}$. We have that σ is well-typed. Moreover, applying our induction hypothesis on Γ', we deduce that $mgu(\Gamma')$ is well-typed, and thus $\{x \mapsto t' mgu(\Gamma')\}$ is well-typed, and $mgu(\Gamma) = mgu(\Gamma') \uplus \{x \mapsto t' mgu(\Gamma')\}$ is well-typed.
- Otherwise, assume that t is a an atom (but not a variable). In such a case, we have that t' is also an atom (and not a variable due to the previous case). Therefore, since t and t' are unifiable, we have that $t = t'$, and $mgu(\Gamma) = mgu(\Gamma')$. Since $\|\Gamma'\| < \|\Gamma\|$, we have that $mgu(\Gamma')$ is well-typed by induction hypothesis, and this allows us to conclude.
- Now, we assume that $t = \mathsf{f}(t_1, \ldots, t_k)$. In such a case, we have that $t' = \mathsf{f}(t'_1, \ldots, t'_k)$ since t' is not a variable and we know that t and t' are unifiable. We have that $mgu(\Gamma) = mgu(\Gamma'')$ where $\Gamma'' = \Gamma' \uplus \{t_1 = t'_1, \ldots, t_k = t'_k\}$. Note that Γ'' is a set of well-typed equations, $\|\Gamma''\| < \|\Gamma\|$ and thus $mgu(\Gamma'')$ is well-typed by induction hypothesis.

This concludes the proof. □

We further assume the existence of an infinite number of constants in Σ_0 (resp. $\Sigma^{\mathsf{atom}}_{\mathsf{fresh}}$, $\Sigma^{\mathsf{bitstring}}_{\mathsf{fresh}}$) of any type.

Example 3.4. Let's continue our running example, with the processes $\overline{P}^1_{\mathsf{NSL}}$ and $\overline{P}^2_{\mathsf{NSL}}$, as defined in Example 2.12. We consider the structure preserving typing system generated from the set $\mathcal{T}_{\mathsf{NSL}} = \{\tau_{\mathsf{a}}, \tau_{\mathsf{b}}, \tau_{\mathsf{c}}, \tau_{na}, \tau_{nb}, \tau_r, \tau_{sk}, \tau_k\}$ of initial types, and the function δ_{NSL} that associates the expected type to each constant/name ($\delta_{\mathsf{NSL}}(a) = \tau_a$, $\delta_{\mathsf{NSL}}(b) = \tau_b$, *etc.*), and the following type to the variables:

$$\delta_{\mathsf{NSL}}(y_1) = \delta_{\mathsf{NSL}}(y'_1) = \tau_{na}, \text{ and } \delta_{\mathsf{NSL}}(y_2) = \delta_{\mathsf{NSL}}(y_3) = \delta_{\mathsf{NSL}}(y'_2) = \delta_{\mathsf{NSL}}(y'_3) = \tau_r.$$

The goal of our main results is to be able to consider only execution traces that comply with a given type, like this one.

3.2 Type compliance

Our main assumption on the typing of protocols is that any two unifiable encrypted subterms are of the same type. The goal of this part is to state this hypothesis formally. For this, we need to define the notion of *encrypted subterms*.

Among the constructor symbols in Σ_{c}, we distinguish those that are *transparent*. They intuitively correspond to constructors that can be freely opened by the attacker, such as pairs, tuples, or lists. A constructor symbol f of arity n is *transparent* if there exists a term $\mathsf{f}(R^{\mathsf{f}}_1, \ldots, R^{\mathsf{f}}_n) \in \mathcal{T}(\Sigma, \Box)$ such that for any term $t \in \mathcal{T}_0(\Sigma, \Sigma_0^+ \uplus \mathcal{N} \uplus \mathcal{X})$ such that $\mathsf{root}(t) = \mathsf{f}$, we have that $\mathsf{f}(R^{\mathsf{f}}_1, \ldots, R^{\mathsf{f}}_n)\{\Box \to t\}\!\downarrow = t$. We denote C_{f} such a term, and write $C_{\mathsf{f}}[t]$ the term obtained by replacing each occurrence of the hole with t.

We write $ESt(t)$ for the set of *encrypted subterms* of t, i.e. the set of subterms that are not headed by a transparent function.

$$ESt(t) = \{u \in St(t) \mid u \text{ is of the form } \mathsf{f}(u_1, \ldots, u_n) \text{ and } \mathsf{f} \text{ is not transparent}\}$$

Example 3.5. Going back to the signature Σ^{ex} introduced in Example 2.1, the symbols $\langle\ \rangle$ and $\langle\ \rangle^3$ are transparent: the contexts $C_{\langle\ \rangle} = \langle \mathsf{fst}(\Box), \mathsf{snd}(\Box)\rangle$ and $C_{\langle\ \rangle^3} = \langle \mathsf{proj}_1^3(\Box), \mathsf{proj}_2^3(\Box), \mathsf{proj}_3^3(\Box)\rangle^3$ satisfy the requirements.

A configuration $\mathcal{K} = (\mathcal{P}; \phi; \emptyset; i)$ is type-compliant if two unifiable encrypted subterms occurring in $\mathcal{K}$ (i.e. either in $\mathcal{P}$ or in ϕ) have the same type. Formally, we use the definition given in the preliminary result [23], which is similar to the one originally introduced by B. Blanchet and A. Podelski in [15].

Definition 3.6. A configuration $\mathcal{K}$ is type-compliant w.r.t. a typing system $(\mathcal{T}_{\mathsf{init}}, \delta)$ if for every $t, t' \in ESt(\mathcal{K})$ we have that: t and t' unifiable implies that $\delta(t) = \delta(t')$.

Example 3.7. Continuing our running example with $\mathcal{K}_{\mathsf{NSL}} = (P_{\mathsf{NSL}}; \phi_0; \emptyset; 0)$, we have that $\mathcal{K}_{\mathsf{NSL}}$ is *not* type-compliant w.r.t. the typing system given in Example 3.4. Indeed, the encrypted subterms are:

- $t_1 = \mathsf{raenc}(\langle \mathsf{a}, y_1\rangle, \mathsf{pub}(sk_b), y_2)$ and $t_1' = \mathsf{raenc}(\langle \mathsf{c}, y_1'\rangle, \mathsf{pub}(sk_a), y_2')$;
- $t_2 = \mathsf{raenc}(\langle y_1, \langle n_b, \mathsf{b}\rangle\rangle, \mathsf{pub}(sk_a), r_2)$ and $t_2' = \mathsf{raenc}(\langle y_1', \langle n_b', \mathsf{a}\rangle\rangle, \mathsf{pub}(sk_c), r_2')$;
- $t_3 = \mathsf{raenc}(n_b, \mathsf{pub}(sk_b), y_3)$ and $t_3' = \mathsf{raenc}(n_b', \mathsf{pub}(sk_a), y_3')$;
- $t_A = \mathsf{pub}(sk_a)$ and $t_B = \mathsf{pub}(sk_b)$.

Actually, we have that t_2 and t_1' are unifiable with $\sigma = \{y_1 \to \mathsf{c},\ y_1' \to \langle n_b, \mathsf{b}\rangle,\ y_2' \to r_2\}$, but we have that:

- $\delta_{\mathsf{NSL}}(t_2) = \mathsf{raenc}(\langle \tau_{na}, \langle \tau_{nb}, \tau_{\mathsf{b}}\rangle\rangle, \mathsf{pub}(\tau_{sk}), \tau_r)$, whereas
- $\delta_{\mathsf{NSL}}(t_1') = \mathsf{raenc}(\langle \tau_{\mathsf{c}}, \tau_{na}\rangle, \mathsf{pub}(\tau_{sk}), \tau_r)$.

Actually, $\mathcal{K}_{\mathsf{NSL}}$ is type-compliant when considering a typing system such that $\delta_{\mathsf{NSL}}(y_1') = \langle \tau_{nb}, \tau_b\rangle$, $\delta_{\mathsf{NSL}}(y_1) = \tau_c$, and keeping the others elements as defined in the typing system introduced in Example 3.4. Note that, w.r.t. this typing system, the attack trace tr given in Example 2.7 is well-typed
.

Consider now the variant P'_{NSL}, as sketched in Example 2.12, where a triple is used instead of two pairs, that is replacing messages t_2 and t_2' by $s_2 = \mathsf{raenc}(\langle y_1, n_b, \mathsf{b}\rangle^3, \mathsf{pub}(sk_a), r_2)$ and $s_2' = \mathsf{raenc}(\langle y_1', n_b', \mathsf{a}\rangle^3, \mathsf{pub}(sk_c), r_2')$. Then the corresponding configuration $\mathcal{K}'_{\mathsf{NSL}} = (P'_{\mathsf{NSL}}; \phi_0; \emptyset; 0)$*is* type-compliant w.r.t. the typing system given in Example 3.4.

3.3 Reduction results

Our main result consists in showing that whenever there is an attack, then there is an attack that is well-typed. This holds for reachability properties as well as equivalence properties.

3.3.1 Reachability. We first prove that for any execution trace, there is a well-typed execution that follows the same sequence of input and output, on the same channels. Formally, we define $\overline{\mathsf{tr}}$ obtained from tr by replacing any action $\mathsf{in}(c, R)$ by $\mathsf{in}(c, \_)$, any $\mathsf{out}(c, \mathsf{w})$ by $\mathsf{out}(c, \_)$, while phase i' is left unchanged. Intuitively, $\overline{\mathsf{tr}}$ only remembers the type of actions, and on which channel.

Theorem 3.8. *Let $\mathcal{K}_P$ be a Σ_0^--configuration type-compliant w.r.t. $(\mathcal{T}_0, \delta_0)$. If $\mathcal{K}_P \overset{\mathsf{tr}}{\Longrightarrow} (\mathcal{P}; \phi; \sigma; i)$ w.r.t. Σ_0^- then there exists a well-typed execution $\mathcal{K}_P \overset{\mathsf{tr}'}{\Longrightarrow} (\mathcal{P}; \phi'; \sigma'; i)$ w.r.t. Σ_0^+ such that $\overline{\mathsf{tr}'} = \overline{\mathsf{tr}}$.*

Conversely, if $\mathcal{K}_P \overset{\mathsf{tr}'}{\Longrightarrow} (\mathcal{P}; \phi'; \sigma'; i)$ is a well-typed execution w.r.t. Σ_0^+, then there exists $\mathcal{K}_P \overset{\mathsf{tr}}{\Longrightarrow} (\mathcal{P}; \phi; \sigma; i)$ w.r.t. Σ_0^- such that $\overline{\mathsf{tr}} = \overline{\mathsf{tr}'}$.

This shows that for any property that can be expressed as a reachability property, it is sufficient to consider well-typed attacks. For example, secrecy of a data s can easily be encoded by adding a witness process of the form $\mathsf{in}(c, s).\mathsf{out}(c_{\mathsf{secret-violated}}, s)$. Then the secret of s is preserved if and only if there is no trace that contains $c_{\mathsf{secret-violated}}$. Similarly, we can consider any property that expresses that some state should never be reached. The second part of Theorem 3.8 can be easily established by replacing constants from Σ_{fresh} (used in the witness) by "fresh" constants from Σ_0^- preserving their sort. The resulting trace is still a valid execution trace. The first and main part of this theorem is proved in Section 4.

3.3.2 Equivalence. Similarly to reachability, we show that whenever two processes are not in trace equivalence, then there is a well-typed witness of non equivalence. Actually, we prove this result for the notion of trace inclusion given in Definition 2.10. Formally, assume given two Σ_0^--configurations $\mathcal{K}_P$ and $\mathcal{K}_Q$ such that $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$ w.r.t. Σ_0^-. A *witness of non-inclusion* is a trace $(\mathsf{tr}, \phi) \in \mathsf{trace}_{\Sigma_0^-}(\mathcal{K}_P)$ such that:

- either there is no ψ such that $(\mathsf{tr}, \psi) \in \mathsf{trace}_{\Sigma_0^-}(\mathcal{K}_Q)$;
- or $\phi \not\sqsubseteq_s \psi$ w.r.t. Σ_0^- for any ψ such that $(\mathsf{tr}, \psi) \in \mathsf{trace}_{\Sigma_0^-}(\mathcal{K}_Q)$.

Note that when a configuration is determinate, once the sequence tr is fixed, there is a unique frame reachable through tr up to static equivalence.

Theorem 3.9. *Let $\mathcal{K}_P$ be a Σ_0^--configuration type-compliant w.r.t. $(\mathcal{T}_0, \delta_0)$ and $\mathcal{K}_Q$ be another Σ_0^--configuration. We have that $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$ w.r.t. Σ_0^- with witness (tr, ϕ) if, and only if, there exists a witness $(\mathsf{tr}', \phi') \in \mathsf{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ of this non-inclusion such that its underlying execution $\mathcal{K}_P \overset{\mathsf{tr}'}{\Longrightarrow} (\mathcal{P}; \phi'; \sigma; i)$ w.r.t. Σ_0^+ is well-typed and $\overline{\mathsf{tr}} = \overline{\mathsf{tr}'}$.*

The main part of the theorem is proved in Section 5. Regarding the fact that the existence of a witness w.r.t. Σ_0^+ can be turned into a witness w.r.t. Σ_0^-, the proof can be done as in the reachability case. This result can be easily established by replacing any symbol from Σ_{fresh} by a symbol of Σ_0^- preserving sorts.

4 TYPING RESULT FOR REACHABILITY

The goal of this section is to provide the main ingredients of the proof of Theorem 3.8. Some of the lemmas are also useful for the proof of Theorem 3.9 on equivalence.

We first define a notion of simple recipes and we explain how we can restrict ourselves to simple recipes (see Section 4.1). Second, our proof crucially relies on a crafted measure on recipes that we introduce in Section 4.2. A witness of reachability, minimal w.r.t. this measure, will be shown to satisfy some good properties, used to derive a well-typed trace.

4.1 Some preliminaries

We introduce the notion of *forced normal form*, denoted $u{\downarrow}$. This is the normal form obtained when applying rewrite rules as soon as the destructor and the constructor match, for example decrypting a message even with a wrong key. Formally, we define the forced rewriting system associated to a set $\mathcal{R}$ of rewriting rules.

Definition 4.1. Given a rewriting rule of the form $\ell_{\mathsf{des}} \to r_{\mathsf{des}}$ as defined in Section 2.1, its associated forced rewriting rule is $\ell'_{\mathsf{des}} \twoheadrightarrow r_{\mathsf{des}}$ where ℓ'_{des} is obtained from ℓ_{des} by keeping only the path to r_{des} in ℓ_{des}. Formally, ℓ'_{des} is defined as follows:

(1) $\ell'_{\mathsf{des}} = \mathsf{des}(x_1, \ldots, x_n)$ when r_{des} is a ground term;
(2) otherwise denoting p_0 the unique position of ℓ_{des} such that $\ell_{\mathsf{des}}|_{p_0} = r_{\mathsf{des}}$ and letting $p_0 = 1.p'_0$, we have that ℓ'_{des} is the linear term such that:

- for any position p' prefix of p_0, we have that $\mathsf{root}(\ell'_{\mathsf{des}}|_{p'}) = \mathsf{root}(\ell_{\mathsf{des}}|_{p'})$;
- $\ell'_{\mathsf{des}}|_{p_0} = r_{\mathsf{des}}$;
- for any other position p' of ℓ'_{des}, we have that $\ell'_{\mathsf{des}}|_{p'}$ is a variable.

We may note that the forced rewriting system associated to a rewriting system as defined in Section 2.1 is well-defined. In particular, given a rewriting rule $\mathsf{des}(t_1, \ldots, t_n) \to r_{\mathsf{des}}$ such that r_{des} is a non ground term, there exists a unique position p'_0 in t_1 such that $t_1|_{p'_0} = r_{\mathsf{des}}$. This comes from the fact that $r_{\mathsf{des}} \in St(t_1)$, and the variable occurring in r_{des} has a unique occurrence in t_1 which is a linear term.

Example 4.2. Going back to our running example, we have that the forced rewriting system $\mathcal{R}_f^{\mathsf{ex}}$ associated to $\mathcal{R}^{\mathsf{ex}}$ is:

$$\mathsf{radec}(\mathsf{raenc}(x, y_1, z), y_2) \twoheadrightarrow x \qquad \mathsf{fst}(\langle x, y\rangle) \twoheadrightarrow x \qquad \mathsf{snd}(\langle x, y\rangle \twoheadrightarrow y$$

Regarding symmetric encryption and signature as introduced in Example 2.4, we get:

$$\mathsf{sdec}(\mathsf{senc}(x, y_1), y_2) \twoheadrightarrow x \qquad \mathsf{getmsg}(\mathsf{sign}(x, y)) \twoheadrightarrow x \qquad \mathsf{check}(x_1, x_2) \twoheadrightarrow \mathsf{ok}$$

Then, given a set $\mathcal{R}_f$ of rewriting rules, a term u can be rewritten to v using $\mathcal{R}_f$ if there is a position p in u, and a rewriting rule $\mathrm{g}(t_1, \ldots, t_n) \twoheadrightarrow t$ in $\mathcal{R}_f$ such that $u|_p = \mathrm{g}(t_1, \ldots, t_n)\theta$ for some substitution θ, and $v = u[t\theta]_p$. As usual, we denote $\twoheadrightarrow^*$, the reflexive-transitive closure of $\twoheadrightarrow$. We may note that such a rewriting system is confluent as it terminates and has no critical pair. As usual, the normal form of a term u is denoted $u{\downarrow}$.

The forced rewriting system allows more rewriting steps than the original one. We will apply it on recipes to simplify them and avoid detours. The following lemma ensures that the term deduced (in a given frame ϕ) through the recipe R would be the same as the one deduced relying on $R{\downarrow}$ as soon as we know that $R\phi{\downarrow}$ is a message.

Lemma 4.3. *Let ϕ be a Σ_0-frame, R a Σ_0-recipe such that $R\phi{\downarrow}$ is a Σ_0-message, and R' be such that $R \twoheadrightarrow R'$. We have that R' is a Σ_0-recipe, and $R'\phi{\downarrow} = R\phi{\downarrow}$.*

In our development, we will consider recipes that have a simple form: they are built using constructor symbols on top of recipes that necessarily extract a subterm of the frame (roughly a recipe made of destructors).

Definition 4.4. A Σ_0-recipe R is a *subterm Σ_0-recipe* if for any Σ_0-frame ϕ such that $R\phi{\downarrow}$ is a Σ_0-message, we have that $R\phi{\downarrow} \in St(\phi)$. We say that R is a *simple Σ_0-recipe* if $R = C[R_1, \ldots, R_k]$ for some context C built using symbols from $\Sigma_{\mathrm{c}} \uplus \Sigma_0$, and each R_i is a subterm Σ_0-recipe such that $\mathsf{root}(R_i) \notin \Sigma_{\mathrm{c}}$.

Simple recipes can be obtained through normalization w.r.t. our forced rewriting system.

Lemma 4.5. *Let θ be a substitution with $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$ and whose image contains Σ_0^--recipes. Let R be a Σ_0^+-recipe in normal form w.r.t. $\twoheadrightarrow$ such that $(R\theta)\phi{\downarrow}$ is a Σ_0^+-message for some Σ_0^--frame ϕ. We have that R' is a simple Σ_0^+-recipe for any $R' \in St(R)$.*

4.2 Our measure

One key step of the proof is to design a measure that reflects how to transform a trace into a well-typed one. Our transformation will proceed by modifying the recipes used by the attacker to forge messages: instead of sending arbitrarily large messages, he should send only small, well-typed messages. The transformation depends on the trace and the frame ϕ_S under consideration and therefore our measure is parameterized by ϕ_S. Finally, the measure of a recipe will be determined by three elements, in lexicographic order.

- First, the size of the term computed by R, that is $R\phi_S\downarrow$;
- Second, the recipe R should be headed by as much constructor terms as possible (In particular, we will prefer the recipe $\langle \mathsf{fst}(\mathsf{w}), \mathsf{snd}(\mathsf{w})\rangle$ over w itself);
- Finally, the size of R itself.

The rest of this section is devoted to the definition of our measure and the establishment of a couple of its properties.

Given a term $t \in \mathcal{T}(\Sigma, \Sigma_0^+ \uplus \mathcal{N})$, we denote $\mathsf{Multi}(t)$, the multiset of elements from $\Sigma \uplus \Sigma_0^+ \uplus \mathcal{N}$ defined as follows:

- $\mathsf{Multi}(a) = \{a\}$ when $a \in \Sigma_0^+ \uplus \mathcal{N}$, and
- $\mathsf{Multi}(\mathsf{f}(t_1, \ldots, t_n)) = \{\mathsf{f}\} \uplus \mathsf{Multi}(t_1) \uplus \ldots \mathsf{Multi}(t_n)$ when $\mathsf{f} \in \Sigma$.

Given a set D of data and a term $t \in \mathcal{T}(\Sigma, D)$, the size of t, denote $|t|$, is the number of function symbols occurring in it. The hat of t is the constructor context R (i.e. a term built on Σ_c with some holes) such that $t = R[t_1, \ldots, t_n]$ with $\mathsf{root}(t_1), \ldots, \mathsf{root}(t_n) \notin \Sigma_c$. We denote it $\mathsf{hat}(t)$.

Given a Σ_0^+-frame ϕ_S together with an ordering $\prec$ on Σ_{fresh} (typically the one corresponding to the order of appearance of these constants in the underlying trace tr_S associated to ϕ_S), the measure $\mu\phi_S$ associated to a Σ_0^+- recipe R is defined as follows (using the lexicographic ordering):

(1) $\mu^1_{\phi_S}(R) = \mathsf{Multi}(R\phi_S\downarrow)$ where elements of the multisets are ordered as follows ($c_{\mathsf{min}} \in \Sigma_0^-$):

$$c_{\mathsf{min}} < \Sigma_0^- \setminus \{c_{\mathsf{min}}\} \uplus \Sigma_c < \Sigma_{\mathsf{fresh}} < \Sigma_d$$

and for elements in Σ_{fresh}, we have that $c < c'$ when $c \prec c'$.

(2) $\mu^2_{\phi_S}(R) = |R\phi_S\downarrow| - |\mathsf{hat}(R)|$.

(3) $\mu^3(R) = |R|$.

Note that $c_{\mathsf{min}} \in \Sigma_0^-$ is the minimal Σ_0^--recipe according to this measure.

We start by establishing some properties regarding this measure. First, we may note that the measure $\mu^2_{\phi_S}$ remains positive since, intuitively, the hat of a term can never disappear by rewriting. (this result is formally stated and proved in Appendix A.2 - see Lemma A.1).

Then we can show that if a recipe R_2 is greater than a recipe R_1 and does not yield a message, then $R_0[R_2]$ is greater than $R_0[R_1]$. This technical lemma will be used very often in our proofs, when reasoning about some reduction that failed inside a bigger term.

Lemma 4.6. *Let $\prec$ be an ordering on Σ_{fresh}, ϕ_S be a Σ_0^+-frame, R_1, R_2 be two Σ_0^+-recipes such that $R_2\phi_S\downarrow$ is not a Σ_0^+-message, and $\mu^1_{\phi_S}(R_1) < \mu^1_{\phi_S}(R_2)$. Let R_0 be a Σ_0^+-recipe, and p a position in R_0. We have that:*

$$\mu^1_{\phi_S}(R_0[R_1]_p) < \mu^1_{\phi_S}(R_0[R_2]_p).$$

The proof of Lemma 4.6 is given in Appendix A.2. It relies on the fact that $R_0[R_1]$ may only reduce more than $R_0[R_2]$.

The measure decreases when applying forced reduction.

Lemma 4.7. *Let $\prec$ be an ordering on Σ_{fresh}, ϕ_S be a Σ_0^+-frame, and R, R' be two Σ_0^+-recipes such that $R \twoheadrightarrow R'$. We have that $\mu_{\phi_S}(R') < \mu_{\phi_S}(R)$.*

The proof of Lemma 4.7 is given in Appendix A.2. Intuitively, if $R \twoheadrightarrow R'$ then we consider the instantiated rule $\ell_{\mathsf{des}}\theta \twoheadrightarrow r_{\mathsf{des}}\theta$ that has been applied. Either $\ell_{\mathsf{des}}\theta\phi_S\downarrow$ is a message. Then both R and R' yield the same term, that is $R\phi_S\downarrow = R'\phi_S\downarrow$. The two first items of the measure are unchanged and we conclude thanks to the third item since R' is smaller than R. In case $\ell_{\mathsf{des}}\theta\phi_S\downarrow$ is not a message, we conclude by Lemma 4.6.

4.3 Completeness

We are now ready to prove the core part of Theorem 3.8.

THEOREM 3.8. *Let $\mathcal{K}_P$ be a Σ_0^--configuration type-compliant w.r.t. $(\mathcal{T}_0, \delta_0)$. If $\mathcal{K}_P \overset{\mathsf{tr}}{\Longrightarrow} (\mathcal{P}; \phi; \sigma; i)$ w.r.t. Σ_0^- then there exists a well-typed execution $\mathcal{K}_P \overset{\mathsf{tr}'}{\Longrightarrow} (\mathcal{P}; \phi'; \sigma'; i)$ w.r.t. Σ_0^+ such that $\overline{\mathsf{tr}'} = \overline{\mathsf{tr}}$.*

Conversely, if $\mathcal{K}_P \overset{\mathsf{tr}'}{\Longrightarrow} (\mathcal{P}; \phi'; \sigma'; i)$ is a well-typed execution w.r.t. Σ_0^+, then there exists $\mathcal{K}_P \overset{\mathsf{tr}}{\Longrightarrow} (\mathcal{P}; \phi; \sigma; i)$ w.r.t. Σ_0^- such that $\overline{\mathsf{tr}} = \overline{\mathsf{tr}'}$.

Given a trace $\mathcal{K}_P \overset{\mathsf{tr}}{\Longrightarrow} (\mathcal{P}; \phi; \sigma; i)$, we need to build a well-typed trace $\mathcal{K}_P \overset{\mathsf{tr}_S}{\Longrightarrow} (\mathcal{P}; \phi_S; \sigma_S; i)$ such that $\overline{\mathsf{tr}_S} = \overline{\mathsf{tr}}$. This is done inductively in the number of execution steps, by modifying the recipes used by the adversary: given a recipe R, some parts of R will be replaced by fresh constants from Σ_{fresh}. Roughly, we will show that tr is actually an instance of tr_S in which the fresh constants of tr_S are replaced by recipes over the current frame ϕ_S.

We first note that once the recipes are fixed (by θ in the following lemma), the underlying terms - computed by the recipes - are entirely determined.

LEMMA 4.8. *Let ϕ_S be a Σ_0^+-frame together with $\prec$ a total ordering on $dom(\phi_S)$. Let θ be a substitution such that $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$, and for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in ϕ_S we have that $c \in dom(\theta)$ and $c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S))$. Moreover, we assume that in case $c \in \Sigma_{\mathsf{fresh}}$ occurs in $\mathsf{w}\phi_S$, then $\mathsf{w}' \prec \mathsf{w}$ for any $\mathsf{w}' \in vars(c\theta)$. We consider the substitution λ whose domain is $dom(\theta)$, and such that:*

$$c\lambda = (c\theta)(\phi_S\lambda)\!\downarrow \text{ for any } c \in dom(\lambda).$$

The substitution λ is well-defined. Moreover, if $(c\theta)\phi_S\!\downarrow$ is a Σ_0^+-message for each $c \in dom(\theta)$, and $(c\theta)\phi_S\!\downarrow$ is an atomic Σ_0^+-message when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$, then $c\lambda$ is a Σ_0^--message for each $c \in dom(\lambda)$, and $c\lambda$ is an atomic Σ_0^--message when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$.

We call λ the first-order substitution associated to θ through ϕ_S.

The fact that λ is well-defined comes from the recursive application of the recipes defined by θ, thanks to the order on the variables. This lemma is formally proved in Appendix A.3.

The relation $c\lambda = (c\theta)(\phi_S\lambda)\!\downarrow$ established by Lemma 4.8 can be generalized to arbitrary recipes R.

LEMMA 4.9. *Let ϕ_S be a Σ_0^+-frame together with $\prec$ a total ordering on $dom(\phi_S)$. Let θ be a substitution such that $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$, and for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in ϕ_S we have that $c \in dom(\theta)$ and $c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S))$. Moreover, we assume that in case $c \in \Sigma_{\mathsf{fresh}}$ occurs in $\mathsf{w}\phi_S$, then $\mathsf{w}' \prec \mathsf{w}$ for any $\mathsf{w}' \in vars(c\theta)$. Let λ be the first-order substitution associated to θ through ϕ_S.*

Assume that for any $c \in dom(\lambda)$, we have that $c\lambda$ is a Σ_0^--message. Moreover, $c\lambda$ is an atomic Σ_0^--message when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$. Let $R \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\theta) \uplus dom(\phi_S))$ such that $R\phi_S\!\downarrow$ is a Σ_0^+-message. We have that $(R\theta)(\phi_S\lambda)\!\downarrow = (R\phi_S\!\downarrow)\lambda$.

The proof follows from an induction on R. The base case is ensured by Lemma 4.8. A formal proof is given in Appendix A.3.

We also note that execution traces do not introduce new encrypted subterms: they are all instances of the initial encrypted subterms.

LEMMA 4.10. *Let $\mathcal{K}_0 = (\mathcal{P}_0; \phi_0; \emptyset; 0)$ be an initial Σ_0-configuration and $\mathcal{K} = (\mathcal{P}; \phi; \sigma; i)$ be a Σ_0-configuration such that $\mathcal{K}_0 \overset{\mathsf{tr}}{\Longrightarrow} \mathcal{K}$ for some tr w.r.t. Σ_0.*

(1) We have that $ESt(\mathcal{K}\sigma) \subseteq ESt(\mathcal{K}_0\sigma)$.

(2) Moreover, in case σ is an mgu between pairs of terms occurring in $ESt(\mathcal{K}_0)$, then we have that $ESt(\mathcal{K}\sigma) \subseteq ESt(\mathcal{K}_0)\sigma$.

The first property follows from the definition: the processes of $\mathcal{K}$ are included in those of $\mathcal{K}_0$ and the output terms stored in the frame ϕ of $\mathcal{K}$ appear initially in the processes of $\mathcal{K}_0$. The second property comes from the fact that unification does not create new encrypted subterm. A formal proof is given in Appendix A.3.

Given $\mathcal{K}_0 = (\mathcal{P}_0; \phi_0; \emptyset; \emptyset)$ be an initial Σ_0^--configuration and $(\mathrm{tr}_S, \phi_S) \in \mathrm{trace}_{\Sigma_0^+}(\mathcal{K}_0)$, we consider the order $\prec$ induced by tr_S. It is defined on the subset of $\mathcal{W} \uplus \Sigma_{\mathsf{fresh}}$ that correspond to elements that occur in tr_S and ϕ_S as follows:

$$u \prec v \;\Leftrightarrow\; \begin{cases} \text{either } u \in dom(\phi_0) \text{ and } v \notin dom(\phi_0) \\ \text{or } u \text{ occurs in } \mathrm{tr}_S \text{ before the first occurrence of } v \text{ in } \mathrm{tr}_S. \end{cases}$$

We are now ready to state how we transform a trace (tr, ϕ) into a well-typed trace (tr_S, ϕ_S).

Proposition 4.11. *Let $\mathcal{K}_P = (\mathcal{P}_0; \phi_0; \emptyset; i_0)$ be an initial Σ_0^--configuration, and $(\mathrm{tr}, \phi) \in \mathrm{trace}_{\Sigma_0^-}(\mathcal{K}_P)$ with underlying substitution σ. Then, there exists $(\mathrm{tr}_S, \phi_S) \in \mathrm{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ with underlying substitution σ_S with $dom(\sigma_S) = dom(\sigma)$ such that $\sigma_S = mgu(\Gamma)\rho$, as well as two substitutions λ and θ such that:*

- $\Gamma = \{(u, v) \mid u, v \in ESt(\mathcal{K}_P) \text{ such that } u\sigma = v\sigma\}$.
- *ρ is a bijective renaming from variables in $dom(\sigma) \setminus dom(mgu(\Gamma))$ to constants in Σ_{fresh} such that $x\rho \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$ if, and only if, $x\sigma$ is an atomic Σ_0^--message.*
- *$dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$, for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in tr_S, we have that $c \in dom(\theta)$, $c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S))$, and $\mathrm{w}' \prec c$ for any $\mathrm{w}' \in vars(c\theta)$ (where $\prec$ is the ordering induced by tr_S).*
- *for any $c \in dom(\theta)$, $(c\theta)\phi_S{\downarrow}$ is a Σ_0^+-message and it is an atom when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$.*
- *λ is the first-order substitution associated to θ through ϕ_S.*
- *$\phi = \phi_S\lambda$, $\sigma = \sigma_S\lambda$, and $(\mathrm{tr}_S\theta)\phi{\downarrow} = \mathrm{tr}\phi{\downarrow}$.*

We say that (ρ, λ, θ) is a well-typed concretization *of (tr_S, ϕ_S) w.r.t. (tr, ϕ) in the context $\mathcal{K}_P$ when ρ, λ, θ satisfy the 6 conditions listed above.*

The fact that (tr_S, ϕ_S) is well-typed is ensured by $\sigma_S = mgu(\Gamma)\rho$. Indeed, since Γ is a set of unifiable encrypted subterms of the protocol, we know that $mgu(\Gamma)$ is well-typed (by assumption on the protocol). Hence σ_S is well-typed. Therefore Theorem 3.8 is a direct consequence of Proposition 4.11 (the details are provided in Appendix A.3).

The proof of Proposition 4.11 is the key step for proving the existence of a well-typed trace. We provide here a detailed sketch of proof while the full proof can be found in Appendix A.3. We omit here the subtleties between Σ_0^+ and Σ_0^-. Proposition 4.11 is proved by induction. Consider an execution trace

$$\mathcal{K}_P \xrightarrow{\mathrm{tr}^-}{}^{*} (\mathcal{P}; \phi^-; \sigma^-; i^-) \xrightarrow{\alpha} (\mathcal{P}; \phi; \sigma; i)$$

By induction, there exist ϕ_S^-, and σ_S^- such that

$$\mathcal{K}_P \xrightarrow{\mathrm{tr}_S^-}{}^{*} (\mathcal{P}; \phi_S^-; \sigma_S^-; i^-)$$

with corresponding λ^-, and θ^- as specified in the proposition. We consider the transition α.

- The case of a phase is immediate: $(\mathrm{tr}_S^-, \phi_S^-)$ can be easily extended.
- The case of an output, i.e. $\alpha = \mathsf{out}(c, \mathrm{w})$ with a corresponding process $\mathsf{out}(c, u).P$ ready to emit, is relatively simple. We simply need to guarantee that $u\sigma_S^-$ is a message, which follows from the fact that $u\sigma^-$ is a message and $\sigma^- = \sigma_S^-\lambda^-$.
- The difficult case is the input case: $\alpha = \mathsf{in}(c, R)$, with a corresponding process $\mathsf{in}(c, u).P$ ready to receive. We have that $R\phi^-{\downarrow} = u\sigma$. We are looking for R_S such that

$$(R_S\theta^-)\phi^-{\downarrow} = u\sigma$$

Such a R_S exists since we could take R. We consider the minimal R_S, w.r.t. our measure, that satisfies this property.

Step 1 We first prove that $R_S\phi_S\downarrow$ is a message: if this is not the case, we show that this contradicts the minimality of R_S w.r.t. the measure.

Step 2 We then show that $R_S = C[R_1, \ldots, R_n]$ where C is a context of constructors and $R_i\phi_S\downarrow$ are encrypted subterms. Indeed, if $R_i\phi_S\downarrow$ was headed by a transparent function, we could push it into the context C and obtain a smaller R_S (w.r.t. item 2 of the measure). In other words, we show that R_S is a simple recipe.

Step 3 R_S still does not satisfy the requirements of Proposition 4.11. It could still be a "big" recipe, that does not satisfy the relation

$$R_S\phi_S\downarrow = u\sigma_S$$

needed to pass the input action. Intuitively, we build $\overline{R_S}$ from R_S by "cutting" the parts that go beyond $u\sigma_S$. By Lemma 4.9, we have that $(R_S\phi_S\downarrow)\lambda = u\sigma_S\lambda$. Assume that $R_S\phi_S\downarrow \neq u\sigma_S$. We know that there exists a position p defined both in $R_S\phi_S\downarrow$ and $u\sigma_S$ where the terms differ. Moreover, since $(R_S\phi_S\downarrow)\lambda = u\sigma_S\lambda$, we actually have that *(i)* either $R_S\phi_S\downarrow|_p \in \Sigma_{\mathsf{fresh}}$; *(ii)* or $u\sigma_S|_p \in \Sigma_{\mathsf{fresh}}$. We distinguish two cases:

- either p is a position in the context C, then we simply cut this part from the context C.
- or p in under the context C (inside some $R_i\phi_S\downarrow$). Then, $R_i\phi_S\downarrow$ is an encrypted subterm of ϕ_S which is equal to an encrypted subterm of $u\sigma_S$. By Lemma 4.10, they are subterms of Γ and therefore, thanks to the application of the mgu, they are equal (thus there is no need to "cut").

These three steps are highlighted in the detailed proof.

Example 4.12. To illustrate **Step 3** of the proof sketched above, consider $\mathcal{P} = \{\mathsf{in}(c, \langle x_1, x_2\rangle).P\}$ (here $u = \langle x_1, x_2\rangle$), and we assume that $x_i\sigma_S = c_i$ with $c_i \in \Sigma_0^+$ (for $i \in \{1, 2\}$). We consider the recipe $R = \langle \mathsf{fst}(\langle a_1, a_2\rangle), \mathsf{senc}(a_1, a_2)\rangle$, with a_1, a_2 in Σ_0^-. We assume that this is the first input, and thus θ^- is the empty substitution. Clearly, we will not choose $R_S = R$ since R is not minimal according to our measure. We will start with $R_S = \langle a_1, \mathsf{senc}(a_1, a_2)\rangle$. We have that $R_S\phi_S\downarrow = \langle a_1, \mathsf{senc}(a_1, a_2)\rangle$ so it is a message, and R_S is a simple recipe: this is consistent with **Step 2**. However, we have that $\langle x_1, x_2\rangle\sigma_S = \langle c_1, c_2\rangle$, and thus according to **Step 3**, R_S is too "big", and we will consider $\overline{R_S} = \langle c_1, c_2\rangle$ with $\theta^+ = \{c_1 \triangleright a_1, c_2 \triangleright \mathsf{senc}(a_1, a_2)\}$.

5 EQUIVALENCE

We now show how to extract a well-typed witness of non trace inclusion $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$ from an arbitrary witness. Compared with the proof for reachability, the proof starts similarly: from a trace $(\mathsf{tr}, \phi) \in \mathcal{K}_P$, we build a well-typed trace $(\mathsf{tr}_S, \phi_S) \in \mathcal{K}_P$. However, an additional work is needed to show that such a tr_S is indeed a witness of non trace inclusion, i.e. either there is no ψ_S such that (tr_S, ψ_S) in $\mathcal{K}_Q$ or $\phi_S \not\sqsubseteq_s \psi_S$ for any ψ_S such that $(\mathsf{tr}_S, \psi_S) \in \mathsf{trace}_{\Sigma_0^-}(\mathcal{K}_Q)$.

5.1 Some preliminaries

We introduce a slightly different notion of static inclusion where the attacker is also given the ability to check whether a message is atomic, i.e. a name in $\mathcal{N}$, a constant in $\Sigma_0^- \uplus \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$, or a constant in Σ of sort atom. This new equivalence actually coincides with the original one.

Definition 5.1. Let ϕ_1, and ϕ_2 be two Σ_0-frames. We write $\phi_1 \sqsubseteq_s^{\mathsf{atom}} \phi_2$ w.r.t. Σ_0 when $dom(\phi_1) = dom(\phi_2)$, and:

- for any Σ_0-recipe R, we have that $R\phi_1\downarrow$ is a Σ_0-message implies that $R\phi_2\downarrow$ is a Σ_0-message;

- for any Σ_0-recipe R, we have that $R\phi_1\downarrow$ is an atomic Σ_0-message, implies that $R\phi_2\downarrow$ is an atomic Σ_0-message; and
- for any Σ_0-recipes R, R' such that $R\phi_1\downarrow$, $R'\phi_1\downarrow$ are Σ_0-messages, we have that: $R\phi_1\downarrow = R'\phi_1\downarrow$ implies $R\phi_2\downarrow = R'\phi_2\downarrow$.

Example 5.2. To illustrate the definition above and its difference with the original definition of static equivalence, we consider Σ^{sign} as described in Example 2.4. Let $\phi = \{\mathsf{w} \triangleright n\}$ and $\psi = \{\mathsf{w} \triangleright \mathsf{hash}(n)\}$ where $n \in \mathcal{N}$.

We have that $\phi \not\sqsubseteq_s^{\mathsf{atom}} \psi$. Indeed, we have that $\mathsf{w}\phi\downarrow$ is an atomic message, whereas $\mathsf{w}\psi\downarrow$ is not. Such a test is not possible when considering $\sqsubseteq_s$. However, relying on a non-linear rule of our rewriting system, here $\mathsf{check}(\mathsf{sign}(x, y), \mathsf{vk}(y)) \to \mathsf{ok}$, we can consider another test that will witness this non-inclusion, namely $\mathsf{check}(\mathsf{sign}(\mathsf{w}, \mathsf{w}), \mathsf{vk}(\mathsf{w})) = \mathsf{ok}$. Indeed such a test holds in ϕ but not in ψ.

More generally, we can show that $\sqsubseteq_s^{\mathsf{atom}}$ coincides with $\sqsubseteq_s$.

Lemma 5.3. *Let ϕ_1 and ϕ_2 be two Σ_0-frames. We have that $\phi_1 \sqsubseteq_s \phi_2$ if, and only if, $\phi_1 \sqsubseteq_s^{\mathsf{atom}} \phi_2$.*

Proof. The proof of the lemma relies on the fact that the attacker may always emulate the atomicity test using one of the non-linear rules, as illustrated in Example 5.2. In case $\phi_1 \sqsubseteq_s^{\mathsf{atom}} \phi_2$, it is easy to see that $\phi_1 \sqsubseteq_s \phi_2$. Therefore, we consider the other implication. Let ϕ_1 and ϕ_2 be two Σ_0-frames such that $\phi_1 \sqsubseteq_s \phi_2$, we have to establish that $\phi_1 \sqsubseteq_s^{\mathsf{atom}} \phi_2$. Let R be a Σ_0-recipe such that $R\phi_1\downarrow$ is an atomic Σ_0-message. We have to show that $R\phi_2\downarrow$ is an atomic Σ_0-message. By hypothesis, we know that there exists $\ell_{\mathsf{des}} \to r_{\mathsf{des}} \in \mathcal{R}$ with ℓ_{des} non-linear, i.e. several occurrences of a variable x with at least one occurring at an atomic position. Let σ be the substitution with $dom(\sigma) = vars(\ell_{\mathsf{des}})$, and $img(\sigma) = \{R\}$, and $R' = \ell_{\mathsf{des}}\sigma$. We have that R' is a Σ_0-recipe such that $R'\phi_1\downarrow$ is a Σ_0-message. Indeed, the rewriting rule applies since the same atomic message occurs at each atomic position.

Relying on our hypothesis $\phi_1 \sqsubseteq_s \phi_2$, we know that both $R\phi_2\downarrow$ and $R'\phi_2\downarrow$ are Σ_0-message. In particular, we have that $(\ell_{\mathsf{des}}\sigma)\phi_2\downarrow$ reduces using $\ell_{\mathsf{des}} \to r_{\mathsf{des}}$ meaning that atomic messages occur at each atomic position, and thus $R\phi_2\downarrow$ is an atomic Σ_0-message. □

Note that this lemma crucially relies on our assumption that our set of rewriting rules contains at least one non-linear rule.

Our measure. We extend our measure to tests, as considered for checking static inclusion. Formally, a test T is either a single recipe R, or a pair R, R' of two recipes. In such a case, i.e. when T is a pair R, R', we define:

$$\mu_{\phi_S}(T) = (\mu^1_{\phi_S}(R) \uplus \mu^1_{\phi_S}(R'),\ \mu^2_{\phi_S}(R) + \mu^2_{\phi_S}(R'),\ \mu^3(R) + \mu^3(R')).$$

5.2 Completeness

The equivalence counterpart of Proposition 4.11 is the following proposition, that states that we can transform a witness of non trace inclusion into a well-typed one.

Proposition 5.4. *Let $\mathcal{K}_P$ and $\mathcal{K}_Q$ be two initial Σ_0^--configurations such that $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$ w.r.t. Σ_0^-. Let $(\mathrm{tr}, \phi) \in \mathsf{trace}_{\Sigma_0^-}(\mathcal{K}_P)$ with underlying substitution σ be a witness of non-inclusion. Then, there exists $(\mathrm{tr}_S, \phi_S) \in \mathsf{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ a witness of this non-inclusion with underlying substitution σ_S such that $\sigma_S = mgu(\Gamma)\rho$, as well as two substitutions λ_P and θ such that $(\rho, \lambda_P, \theta)$ is a well-typed concretization of (tr_S, ϕ_S) w.r.t. (tr, ϕ) in the context $\mathcal{K}_P$, as defined in Proposition 4.11.*

Exactly like for the reachability case, the fact that (tr_S, ϕ_S) is well typed is ensured by $\sigma_S = mgu(\Gamma)\rho$. Indeed, since Γ is a set of unifiable encrypted subterms of $\mathcal{K}_P$, we know that $mgu(\Gamma)$ is

well-typed (by assumption on the protocol). Hence σ_S is well-typed. Therefore Theorem 3.9 is a direct consequence of Proposition 5.4 (the details are provided in Appendix B).

We provide here a detailed sketch of proof of Proposition 5.4, leaving the full proof in Appendix B. Consider $(\mathrm{tr}, \phi) \in \mathrm{trace}_{\Sigma_0^-}(\mathcal{K}_P)$ with underlying substitution σ be a witness of non-inclusion. We explain how to build a well-typed witness of non inclusion. Thanks to Proposition 4.11, there exists a well-typed trace $(\mathrm{tr}_S, \phi_S) \in \mathrm{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ with underlying substitution σ_S such that $\sigma_S = mgu(\Gamma)\rho$, and λ_P and θ satisfying the conditions of Proposition 4.11. We would like to show that (tr_S, ϕ_S) is a witness of non-inclusion. There must exist a trace $(\mathrm{tr}_S, \psi_S) \in \mathrm{trace}_{\Sigma_0^+}(\mathcal{K}_Q)$ such that $\phi_S \sqsubseteq_s \psi_S$, otherwise we can already conclude that (tr_S, ϕ_S) is a witness of non-inclusion. Then the first step of the proof consists in showing that we can instantiate (tr_S, ψ_S) by θ and λ_Q such that $\psi = \psi_S\lambda_Q$, $\mathrm{tr}\psi\downarrow = (\mathrm{tr}_S\theta)\psi\downarrow$, and $(\mathrm{tr}, \psi) \in \mathrm{trace}_{\Sigma_0^-}(\mathcal{K}_Q)$. We also show that ψ satisfies the assumptions of Lemmas 4.8 and 4.9.

In a second step of the proof, we show that $\phi_S \sqsubseteq_s \psi_S$ implies $\phi \sqsubseteq_s \psi$, hence a contradiction. More specifically, we show by induction on $\mu_{\phi_S}(T)$, that:

$$\text{for any test } T, \text{ if } T\theta \text{ holds for } \phi \text{ then it holds for } \psi. \qquad (*)$$

This allows us to conclude that any test T that holds in ϕ also holds in ψ since we may simply consider tests without constants in $dom(\theta)$. Hence $\phi \sqsubseteq_s \psi$.

We consider all the tests defined by the static inclusion $\sqsubseteq_s^{\mathsf{atom}}$. In particular, the attacker can test directly whether a message is an atom or not, which avoids to consider a "big" recipe of the form $\mathsf{check}(\mathsf{sign}(R, R), \mathsf{vk}(R)) = \mathsf{ok}$ instead of simply R. We show (*) by induction on our measure. Therefore let's assume that (*) holds for any test T' such that $\mu_{\phi_S}(T') < \mu_{\phi_S}(T)$. We consider the three possible cases for T.

- Either T checks whether the term induced by R is a message. We need to show that $(R\theta)\phi\downarrow$ is a message implies $(R\theta)\psi\downarrow$ is a message. If $R\phi_S\downarrow$ is a message then so is $R\psi_S\downarrow$ by static inclusion and we can rather easily conclude that $(R\theta)\psi\downarrow$ is a message. If $R\phi_S\downarrow$ is not a message, similarly to the reachability case, we can build smaller tests, apply the induction hypothesis and reconstruct R. An additional difficulty comes from the fact that we now also need to transfer the properties on ψ.
- Or T checks whether the term induced by R is atomic. We need to show that $(R\theta)\phi\downarrow$ is atomic implies $(R\theta)\psi\downarrow$ is atomic. Thanks to the previous case, we already know that $(R\theta)\psi\downarrow$ is a message. Actually, $R\phi_S\downarrow$ must be atomic because $(R\theta)\phi\downarrow = R\phi_S\downarrow\lambda_P$ is atomic and due to the constraints on λ_P. We deduce that $R\psi_S\downarrow$ is atomic by static inclusion, and thus $(R\theta)\psi\downarrow = R\psi_S\downarrow\lambda_Q$ is atomic since λ_Q preserves atomicity.
- Or T is an equality test $R = R'$. If R and R' are headed by the same constructor symbol, we may simply open each recipe and apply the induction hypothesis. Therefore, we may assume that $R = C[R_1, \ldots, R_n]$ and R' is headed by a destructor. As for the reachability case, we can show that the R_i as well as R' are subterm recipes yielding an encrypted subterm. We have that $R\phi_S\downarrow$ and $R'\phi_S\downarrow$ are messages. Either $R\phi_S\downarrow = R'\phi_S\downarrow$. Then this holds for ψ_S as well and therefore for ψ. Or $R\phi_S\downarrow \neq R'\phi_S\downarrow$ and yet $(R\phi_S)\downarrow\lambda_P = (R'\phi_S)\downarrow\lambda_P$. Let's consider a leaf position p on which the two terms differ, i.e. p exits in both $R\phi_S\downarrow$ and $R'\phi_S$, but $R\phi_S\downarrow|_p \neq R'\phi_S\downarrow|_p$. We assume that p is a leaf of $R'\phi_S\downarrow$, and since λ_P makes these two terms equal, we know that $R'\phi_S\downarrow = c \in \Sigma_{\mathsf{fresh}}$ (the case where p is a leaf of $R\phi_S$ is actually simpler).
  - either p belongs to one of the $R_i\phi_S\downarrow$. Then, as for the reachability case, $R_i\phi_S\downarrow$ is an encrypted subterm of ϕ_S which is equal to an encrypted subterm of $R'\phi_S\downarrow$, and thus an encrypted subterm of ϕ_S since R' is a subterm recipe. By Lemma 4.10, they are subterms of Γ and therefore, thanks to the application of the mgu, they are equal.

– or p belongs to the context C, then we build $\overline{C}$ from C by replacing $C|_p$ by c for all such c. Consider the corresponding recipe $\overline{R} = \overline{C}[R_1, \ldots, R_n]$. We show that the equality $\overline{R} = R'$ holds in ϕ_S since we have removed all the differences. Therefore we have that $\overline{R} = R'$ holds in ψ_S. Moreover, the equality $\overline{R} = R$ holds relying on our induction hypothesis.

6 DEALING WITH REPLICATED PROCESSES

In this section, we consider the case of processes with replication, that is processes that can be executed an arbitrary number of times. We first extend our process algebra (syntax and semantics). Then, we explain how the type-compliance hypothesis can be effectively checked on replicated processes. Finally, we show how to extend the results stated in Section 3 to this new setting.

6.1 Syntax and semantics

When considering replicated processes, it becomes useful to add a contruct to generate fresh names. Therefore, in addition to the replication operator, we consider two additional instructions allowing us to generate fresh basic names, and fresh channel names. For this purpose, we introduce the set Ch_{fresh} of fresh channels. Those channel names, as every channel name we have considered so far, are public. Then, we extend our process algebra as follows:

$$P, Q := \ldots \mid \mathsf{new}\, n.P \mid \mathsf{new}\, c'.\mathsf{out}(c, c').P \mid !P$$

where $n \in \mathcal{N}$, $c \in Ch \uplus Ch_{\mathsf{fresh}}$ and $c' \in Ch_{\mathsf{fresh}}$.

We extend our semantics with the following 3 rules:

$$\begin{array}{rcll} (i : \mathsf{new}\, n.P \uplus \mathcal{P}; \phi; \sigma; i) & \xrightarrow{\tau} & (i : P\{^m/_n\}; \phi; \sigma; i) & \text{with } m \in \mathcal{N} \text{ fresh.} \\ (i : \mathsf{new}\, c'.\mathsf{out}(c, c').P \uplus \mathcal{P}; \phi; \sigma; i) & \xrightarrow{\mathsf{out}(c, c'')} & (i : P\{^{c''}/_{c'}\} \uplus \mathcal{P}; \phi; \sigma; i) & \text{with } c'' \text{ a fresh channel.} \\ (i : !P \uplus \mathcal{P}; \phi; \sigma; i) & \xrightarrow{\tau} & (i : P' \uplus i : !P \uplus \mathcal{P}; \phi; \sigma; i) & \text{with } P' \text{ a copy of } P \end{array}$$

where variables bound in the inputs are α-renamed.

The construction $\mathsf{new}\, n.P$ generated a fresh name m and replaces n by m in P. The public channel creation instruction $\mathsf{new}\, c'.\mathsf{out}(c, c').P$ creates a fresh channel c'', and replaces c' by c'' in P. The channel name c'' is not added into the frame. Actually, all the channel names are assumed to be public and the label $\mathsf{out}(c, c'')$ on the arrow makes this instruction observable. Finally, $!P$ allows us to launch a new copy of P, and thus consider processes that can be executed an arbitrary number of times.

First, we may note that our typing result stated in Section 3 still apply when extending our process algebra with the two constructions allowing one to generate fresh basic and channel names. We did not include these instructions in our initial model since they are mostly useless in a model without replication. In the remaining, we admit Theorems 3.8 and 3.9 for this simple extension, and we show how to add the replication operator. For this very simple extension, the main point is to assume that the typing system $(\mathcal{T}, \delta)$ under study respects the name binding, that is:

$$(i : \mathsf{new}\, n.P \uplus \mathcal{P}; \phi; \sigma; i) \xrightarrow{\tau} (i : P\{^m/_n\} \uplus \mathcal{P}; \phi; \sigma; i) \quad \text{with } m \in \mathcal{N} \text{ fresh and } \delta(n) = \delta(m)$$

Now, we go back to our replication operator. We say that a process P' is a *finite version* of P if P' is P, except that each subprocess of the form $!Q$ occurring in P is replaced with $(Q_1 \mid \cdots \mid Q_n)$ (for some $n \geq 0$) where each Q_i is a copy of Q (in which bound variables are α-renamed). In presence of nested replications, the transformation may be applied several times until obtaining a process without replication. This notion naturally extends to multisets of processes and to configurations. Intuitively, a finite version represents a version of the process that can only be executed a fixed number of times. We define the k-unfolding $\mathsf{unfold}_k(P)$ of the replicated process P as a finite

version of P such that each replication has been unfolded exactly k times. For instance, we have that $\mathsf{unfold}_1(P)$ is the process obtained from P by simply removing the ! operator whereas $\mathsf{unfold}_0(P)$ is the process obtained from P by removing parts of the process under a replication.

6.2 Type compliance

We now consider a process P in our extended syntax (i.e. with replication), and a typing system $(\mathcal{T}, \delta)$. We assume that $\delta(m) = \delta(n)$ whenever $m \in \mathcal{N}$ instantiates the name n, and $\delta(x) = \delta(y)$ when x and y are an α-renaming of the same variable due to replication. We now establish the following result:

> If $\mathsf{unfold}_2(\mathcal{P})$ is type compliant w.r.t. $(\mathcal{T}, \delta)$, then any finite version of $\mathcal{P}$ is type compliant w.r.t. $(\mathcal{T}, \delta)$, and we say that $\mathcal{P}$ is type-compliant w.r.t. $(\mathcal{T}, \delta)$.

Sketch of proof. Let $\mathcal{P}'$ be a finite version of $\mathcal{P}$, and $t_1, t_2 \in \mathit{ESt}(\mathcal{P}')$ be two unifiable terms. We have to prove that $\delta(t_1) = \delta(t_2)$. Without loss of generality, we may assume that we are in one of the two following cases:

(1) In $\mathcal{P}$, the term t_1 occurs in a prefix of the process where the term t_2 occurs.
(2) There is a $(Q_1 \mid \cdots \mid Q_n)$ in $\mathcal{P}'$ such that t_1 occurs in Q_1 and t_2 occurs in Q_2.

In the first case, let $Q \in \mathcal{P}$ be the process where t_2 occurs. We have that t_1 also occurs in Q. So both t_1 and t_2 occur in Q, so t_1 and t_2 occur in $\mathsf{unfold}_1(Q)$, and thus in $\mathsf{unfold}_2(\mathcal{P})$. As $\mathsf{unfold}_2(\mathcal{P})$ is type-compliant, we have $\delta(t_1) = \delta(t_2)$.

In the second case, we distinguish two cases:

- either $(Q_1 \mid \cdots \mid Q_n)$ is a subprocess of $\mathcal{P}$ and thus t_1, t_2 occur in $\mathcal{P}$ and thus in $\mathsf{unfold}_2(\mathcal{P})$. By hypothesis, we thus have $\delta(t_1) = \delta(t_2)$, and this allows us to conclude.
- or $(Q_1 \mid \cdots \mid Q_n)$ is coming from a $!Q$, and up to a (well-typed) α-renaming of variables, $(Q_1 \mid Q_2) = \mathsf{unfold}_2(!Q)$. Therefore, we have that t_1 and t_2 are encrypted subterms of $\mathsf{unfold}_2(!Q)$, and thus of $\mathsf{unfold}_2(\mathcal{P})$. We get $\delta(t_1) = \delta(t_2)$, and we conclude. □

6.3 Typing result for reachability

Now, we explain how to derive a typing result for reachability for processes with replication. More precisely, we establish the following statement:

> Let $\mathcal{K}_P$ be a Σ_0^--configuration type-compliant w.r.t. $(\mathcal{T}_0, \delta_0)$. If $\mathcal{K}_P \overset{\mathsf{tr}}{\Longrightarrow} (\mathcal{P}; \phi; \sigma; i)$ w.r.t. Σ_0^- then there exists a well-typed execution $\mathcal{K}_P \overset{\mathsf{tr}'}{\Longrightarrow} (\mathcal{P}; \phi'; \sigma'; i)$ w.r.t. Σ_0^+ such that $\overline{\mathsf{tr}'} = \overline{\mathsf{tr}}$.

Remember that $\overline{\mathsf{tr}}$ is obtained from tr by replacing any action $\mathsf{in}(c, R)$ by $\mathsf{in}(c, \_)$, any $\mathsf{out}(c, \mathsf{w})$ by $\mathsf{out}(c, \_)$, while keeping the other actions (i.e. phase i and $\mathsf{out}(c, c'')$) unchanged.

First, we make the observation that every attack is an attack against a finite execution (**Fact 1**). Then, we establish that any execution of a finite version is an execution of the original process (**Fact 2**). These properties correspond to expected behaviors when considering finite versions of a process.

Fact 1. If there is a trace (tr, ϕ) of $\mathcal{P}$ of length at most ℓ, then (tr, ϕ) is a trace of $\mathsf{unfold}_\ell(\mathcal{P})$.

Sketch of proof. We proceed by induction on ℓ. When $\ell = 0$, the result is obvious. Now, assume that $\mathsf{tr} = \mathsf{tr}_0.\alpha$ for some observable action α. Since the induction hypothesis applies on tr_0, we know that the execution corresponding to tr_0 can be done starting from $\mathsf{unfold}_{\ell-1}(\mathcal{P})$. Then either tr is

a trace of $\mathsf{unfold}_\ell(\mathcal{P})$, or there exists a replication in $\mathcal{P}$ and some extra unfoldings are necessary to execute α. Actually, one supplementary unfolding will be enough. Hence the result. □

Fact 2. Let $\mathcal{P}'$ be a finite version of some process $\mathcal{P}$ with replication. If (tr, ϕ) is a trace of $\mathcal{P}'$, then it is also a trace of $\mathcal{P}$.

Sketch of proof. To mimic the execution tr starting from $\mathcal{P}$ (with replication), we will simply unfold the replications when needed inserting some τ (unobservable actions) in the trace tr. □

Now, we are able to establish our typing result for reachability in presence of replicated processes. Assuming we have a trace tr starting from some initial configuration $\mathcal{K}_P = (\mathcal{P}; \emptyset; \emptyset; 0)$. Fact 1 tells us that this trace can be performed starting from $\mathcal{K}'_P = (\mathcal{P}'; \emptyset; \emptyset; 0)$ where $\mathcal{P}' = \mathsf{unfold}_\ell(\mathcal{P})$. Now, Theorem 3.8 (more precisely its simple extension discussed in Section 6.1) applies. Note that type-compliance is satisfied by any finite version of $\mathcal{P}$ and thus in particular $\mathsf{unfold}_\ell(\mathcal{P})$ is type-compliant. Therefore, there exists a well-typed trace tr' of $\mathcal{K}'_P$ such that $\overline{\mathsf{tr}'} = \overline{\mathsf{tr}}$, and Fact 2 allows us to ensure that the trace tr' can be performed starting from the initial configuration $\mathcal{K}_P$. This proves the result.

6.4 Equivalence result

In this section, we explain how to establish the typing result regarding equivalence. We aim at establishing the following statement.

> Let $\mathcal{K}_P$ be a Σ_0^--configuration type-compliant w.r.t. $(\mathcal{T}_0, \delta_0)$ and $\mathcal{K}_Q$ a Σ_0^--configuration. If $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$ w.r.t. Σ_0^- then there exists a witness $(\mathsf{tr}, \phi) \in \mathsf{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ of this non-inclusion such that its underlying execution $\mathcal{K}_P \overset{\mathsf{tr}}{\Longrightarrow} (\mathcal{P}; \phi; \sigma; i)$ w.r.t. Σ_0^+ is well-typed.

Sketch of proof. Let (tr, ϕ) be a witness of $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$ w.r.t. Σ_0^- and we denote ℓ the length of tr. Fact 1 tells us that there is a finite version $\mathcal{K}'_P$ of $\mathcal{K}_P$ (the ℓ^{th} unfolding of the process is actually a good candidate) such that (tr, ϕ) is a trace of $\mathcal{K}'_P$. If there was a ψ such that (tr, ψ) is a trace of $\mathsf{unfold}_\ell(\mathcal{K}_Q)$, then by Fact 2, (tr, ψ) would be a trace of $\mathcal{K}_Q$, and thus $\phi \not\sqsubseteq_s \psi$. So the trace (tr, ϕ) is a trace of non-inclusion for $\mathcal{K}'_P \not\sqsubseteq \mathsf{unfold}_\ell(\mathcal{K}_Q)$.

By Theorem 3.9, there is a well-typed trace (tr', ϕ') of this non-inclusion (because $\mathcal{K}'_P$ is type compliant and $\mathcal{K}'_P$ and $\mathsf{unfold}_\ell(\mathcal{K}_Q)$ are processes without replication). By Fact 2, (tr', ϕ') is a trace of $\mathcal{K}_P$. Now, it only remains to prove that tr' is a trace of non-inclusion for $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$. Thanks to Fact 1 and since tr and tr' have the same length (as $\overline{\mathsf{tr}'} = \overline{\mathsf{tr}}$) if there is a ψ' such that (tr', ψ') of $\mathcal{K}_Q$, then (tr', ψ') is also a trace of $\mathsf{unfold}_\ell(\mathcal{K}_Q)$. As (tr', ϕ') is a witness of non-inclusion for $\mathcal{K}'_P \not\sqsubseteq \mathsf{unfold}_\ell(\mathcal{K}_Q)$, we have that $\phi' \not\sqsubseteq_s \psi'$. This allows us to conclude that (tr', ϕ') is indeed a witness of non-inclusion for $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$.

7 EXAMPLES

We review several protocols of the literature and identify whether our main results can be applied, that is, we identify when the protocols satisfy the type-compliance condition. Our small attack result is particularly interesting when it is established w.r.t. a structure preserving typing system. Indeed, in such a case, the typing result allows one to bound the size of the messages involved in the smallest witness. Therefore, in this section, we aim at establishing type-compliance w.r.t. a structure preserving typing system. We first discuss which scenario is considered.

7.1 Complete scenario

We explain which scenario and which security property we consider, illustrated on the NSL protocol. As presented in Example 2.5 and Example 2.12, we may need to consider a rich scenario

when searching for attacks. In the case of a 2-agent protocol, like the Needham-Schroeder-Lowe protocol, we wish to instantiate the roles P_A and P_B with different combinations of honest and dishonest agents. We consider a and b to be two honest agents, whereas c is dishonest, and denote by $P_A(z_A, z_B)$ (resp. $P_B(z_B, z_A)$) the role P_A (resp. P_B) played by agents z_A and z_B. Then we consider the following:

$$P^{\mathsf{semi}}_{\mathsf{NSL}} = P_A(\mathrm{a},\mathrm{b}) \mid P_B(\mathrm{b},\mathrm{a}) \mid P_A(\mathrm{a},\mathrm{c}) \mid P_B(\mathrm{b},\mathrm{c}) \mid P_A(\mathrm{b},\mathrm{a}) \mid P_B(\mathrm{a},\mathrm{b}) \mid P_A(\mathrm{b},\mathrm{c}) \mid P_B(\mathrm{a},\mathrm{c})$$

$P^{\mathsf{semi}}_{\mathsf{NSL}}$ considers all possible combinations of a, b, c within P_A and P_B, except that we exclude the processes $P_A(\mathrm{c},\mathrm{a})$ and $P_A(\mathrm{c},\mathrm{b})$ as well as $P_B(\mathrm{c},\mathrm{b})$ and $P_B(\mathrm{c},\mathrm{a})$. Indeed, these four processes describe the behavior of a dishonest agent c, which is already included within the semantics of our model. $P^{\mathsf{semi}}_{\mathsf{NSL}}$ models the possible two-way interactions between honest agents, and between honest agents willing to establish a session with an attacker. Note that processes $P_A(\mathrm{a},\mathrm{b})$, $P_B(\mathrm{a},\mathrm{b})$ and $P_B(\mathrm{a},\mathrm{c})$ correspond respectively to P_A, P_B and P'_B in Example 2.5. $P^{\mathsf{semi}}_{\mathsf{NSL}}$ models a complete scenario with one session only of each possible instantiation of a role. As explained in Section 6, type-compliance and well-typedness can be easily lifted to protocols with replication. In a slightly different setting, [23] has established that type compliance is guaranteed for an unbounded number of sessions as soon as a protocol is type-compliant when 2 sessions are considered for each role and each possible choice of agents. For this reason, we consider a complete protocol $P^{\mathsf{complete}}_{\mathsf{NSL}} = P^{\mathsf{semi}}_{\mathsf{NSL}} \mid P'^{\mathsf{semi}}_{\mathsf{NSL}}$ where $P'^{\mathsf{semi}}_{\mathsf{NS}}$ is a copy of $P^{\mathsf{semi}}_{\mathsf{NSL}}$ alpha-renamed to ensure names and variables are all distinct in $P^{\mathsf{complete}}_{\mathsf{NSL}}$, and the same type is given to names and variables that have been alpha-renamed. This scenario is complete in the sense that, according to the approach of [23], a richer scenario derived from this one by adding more copies of a process will still be type-compliant according to our definition (note that no more encrypted subterms will be introduced).

For the security property, we consider the secrecy of a key or a nonce, where secrecy is encoded as a combination of key usability and "which key-concealing". For NSL, we consider secrecy of the nonce n_b as received by $P_A(\mathrm{a},\mathrm{b})$. To that end we consider two variants of $P_A(\mathrm{a},\mathrm{b})$: in the first one, we add a final action $\mathsf{out}(c_A, \mathsf{senc}(m_1, n_b))$, whereas we add $\mathsf{out}(c_A, \mathsf{senc}(m_2, k))$ in the second, where k is a fresh nonce, and m_1 and m_2 are two publicly known constants. However, for NSL, the resulting protocols are not type-compliant. Indeed, there is an encrypted subterm $\mathsf{aenc}(x_1, \mathsf{pub}(sk_b), r_3)$ in P_A (Example 2.5) which can be unified with other encrypted subterm from the protocol. Note that the scenario we consider is richer than the scenario from Example 3.7, which explains why the typing system considered is not enough to make the entire protocol type-compliant. To ensure type-compliance, we need here to consider a tagged version of the protocol where a public constant is appended to each plaintext in the specification. The informal specification is given below.

$$\begin{aligned} A \to B :&\quad \{1, N_a, A\}_{\mathsf{pub}(B)} \\ B \to A :&\quad \{2, N_a, N_b, B\}_{\mathsf{pub}(A)} \\ A \to B :&\quad \{3, N_b\}_{\mathsf{pub}(B)} \end{aligned}$$

7.2 Review of key-exchange protocols

We consider several protocols from the literature and investigate their type-compliance for the complete scenario. As well as for the Needham-Schroeder-Lowe protocol, the security property we consider is the secrecy of secret (nonce or session key) exchanged between two honest agents a and b, encoded as a combination of key usability and "which key-concealing". Tuples are encoded directly, without using nested pairs.

The complete scenario for a 3-party protocol with a trusted server S is somewhat more complex than for a 2-party protocol. We need to consider a server S willing to establish a session between

	Type-compliance with tuples	Type-compliance with tagging
Needham-Schroeder-Lowe	×	✓
Wide Mouth Frog	✓	✓
Denning Sacco with shared keys	✓	✓
Needham-Schroeder with shared keys	✓	✓
Yahalom-Lowe	✓	✓
Yahalom-Paulson	×	✓
Otway-Rees	✓	✓
Denning Sacco with signature	✓	✓
Passive authentication	✓	✓
Active authentication	✓	✓

Fig. 3. Type-compliance of protocols for Sections 7.2 and 7.3.

all pairs of agents in {a, b, c}, in addition of the usual interactions between a, b and c. Each of the protocols is thus modeled as 14 processes, leading to 28 processes after duplication.

Type-compliance of the protocols has been verified automatically. In some cases, we needed to tag the protocol to ensure type compliance, as for the case of the NSL protocol. Our findings are summarized in Figure 3, which describes which protocol is type-compliant, with or without tags. We discuss below each protocol individually.

Wide Mouth Frog. The Wide Mouth Frog protocol can be informally described as follows.

$$\begin{array}{ll} A \to S: & A,\ \{B, K_{ab}\}_{K_{as}} \\ S \to B: & \{A, K_{ab}\}_{K_{bs}} \end{array}$$

Here we verify the strong secrecy of K_{ab} as received by B. We consider a structure-preserving typing system which associates the type agent to agents a, b, c and S, the type session to negotiated keys between the agents such as K_{ab} and the type longterm to long-term keys K_{as}, K_{bs} and K_{cs}. The constants m_1 and m_2 occurring in the security property (as for the NSL protocol) are typed with a type constant. This protocol is type-compliant, without requiring any additional tagging, as every encrypted subterm is of the form $\mathsf{senc}(\langle \mathsf{agent}, \mathsf{session}\rangle, \mathsf{longterm})$, except for the terms that come from the encoding of the security property, which use a constant as plaintext and thus cannot be unified with other ciphertext.

Denning Sacco with shared keys. The Denning Sacco protocol can be informally described as follows.

$$\begin{array}{ll} A \to S: & A, B \\ S \to A: & \{B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}} \\ A \to B: & \{K_{ab}, A\}_{K_{bs}} \end{array}$$

We consider the same typing system as for the Wide Mouth Frog protocol. This protocol is type-compliant, without requiring any additional tagging. The difference in the arity of the plaintexts in the protocol specification ensures all possible unifications between encrypted subterms occur between encrypted subterms of the same type.

Needham-Schroeder with shared keys. The Needham-Schroeder symmetric key protocol can be informally described as follows.

$$\begin{array}{ll} A \to S: & A, B, N_a \\ S \to A: & \{B, N_a, K_{ab}, \{A, K_{ab}\}K_{bs}\}_{K_{as}} \\ A \to B: & \{A, K_{ab}\}_{K_{bs}} \\ B \to A: & \{\mathsf{req}, N_b\}_{K_{ab}} \\ A \to B: & \{\mathsf{rep}, N_b\}_{K_{ab}} \end{array}$$

This protocol is type-compliant, without requiring any additional tagging, using the same structure-preserving typing system as before.

Yahalom-Lowe. The Yahalom-Lowe protocol can be informally described as follows.

$$\begin{array}{ll} A \to B: & A, N_a \\ B \to S: & \{A, N_a, N_b\}_{K_{bs}} \\ S \to A: & \{B, K_{ab}, N_a, N_b\}_{K_{as}} \\ S \to B: & \{A, K_{ab}\}_{K_{bs}} \\ A \to B: & \{A, B, S, N_b\}_{K_{ab}} \end{array}$$

This protocol is type-compliant, without requiring any additional tagging, using the same structure-preserving typing system as before. Note that type-compliance depends on the encoding of tuples: when considered as nested pairs, type-compliance is not guaranteed, as more encrypted subterms become unifiable.

Yahalom-Paulson. The Yahalom-Paulson protocol can be informally described as follows.

$$\begin{array}{ll} A \to B: & A, N_a \\ B \to S: & B, N_b, \{A, N_a\}_{K_{bs}} \\ S \to A: & N_b, \{B, K_{ab}, N_a\}_{K_{as}}, \{A, B, K_{ab}, N_b\}_{K_{bs}} \\ A \to B: & \{A, B, K_{ab}, N_b\}_{K_{bs}}, \{N_b\}_{K_{ab}} \end{array}$$

This protocol is not type-compliant. Indeed its formal specification contains a term $\mathsf{senc}(x_{nb}, x_{Kab})$, in the encrypted subterms of A, which can be unified with any other encrypted subterm from the protocol, such as subterms encrypted with long-term keys K_{as} and K_{bs}. To ensure type-compliance with a non-trivial typing system, we consider a tagged version of the protocol.

$$\begin{array}{ll} A \to B: & A, N_a \\ B \to S: & B, N_b, \{1, A, N_a\}_{K_{bs}} \\ S \to A: & N_b, \{2, B, K_{ab}, N_a\}_{K_{as}}, \{3, A, B, K_{ab}, N_b\}_{K_{bs}} \\ A \to B: & \{3, A, B, K_{ab}, N_b\}_{K_{bs}}, \{4, N_b\}_{K_{ab}} \end{array}$$

Otway-Rees. The Otway-Rees protocol can be informally described as follows.

$$\begin{array}{ll} A \to B: & M, A, B, \{N_a, M, A, B\}K_{as} \\ B \to S: & M, A, B, \{N_a, M, A, B\}K_{as},\ \{N_b, M, A, B\}K_{bs} \\ S \to B: & M, \{N_a, K_{ab}\}K_{as},\ \{N_b, K_{ab}\}K_{bs} \\ B \to A: & M, \{N_a, K_{ab}\}K_{as} \end{array}$$

This protocol is type-compliant, without requiring any additional tagging, using the same structure-preserving typing system as before. As for the Yahalom-Lowe protocol, type-compliance relies here on the direct encoding of tuples, and would not hold with nested pairs. Actually, we can also consider a different typing system which consists of typing the variables used to model ciphertext forwarding using a constant. This does not correspond to the expected type in a normal execution,

but we can show that the protocol is type-compliant. This yields an interesting property w.r.t. attack search. Namely, we deduce that it is useless to instantiate these variables with complex terms when looking for an attack.

Denning Sacco with signature. This protocol, presented in [13], can be seen as a simplified version of the Denning Sacco protocol. It can be can be informally described as follows.

$$\begin{array}{ll} A \to B: & A, B, \{\mathsf{sign}(\langle A, B, k\rangle, skA)\}_{\mathrm{pub}(B)} \\ B \to A: & \{m\}_k \end{array}$$

This protocol is type-compliant, without requiring any additional tagging, using the same structure-preserving typing system as before.

7.3 E-passport protocols

We also consider two authentication protocols used in the e-passport application, namely the passive authentication (PA) protocol and the active authentication (AA) protocol. These protocols enable a reader (agent R in the following informal specifications) to authenticate a passport (agent P). The two agents already possess sessions keys thanks to a prior execution of a key-exchange protocol. For this reason, the security property we want to verify here is a variant of unlinkability: we want to model the inability for an attacker to distinguish between a scenario involving Passport 1 and two sessions of Passport 2, and a scenario involving two copies of Passport 1 and only one session of Passport 2. This models the inability for the attacker to link two sessions of a same passport together. We then investigate the type-compliance of the resulting configurations, and summarize the results in Figure 3. Similarly to the key-exchange protocols, we consider a version of the protocol where each session has been duplicated, to ensure all possible unifications are observed.

Passive authentication. This protocol can be informally described as follows.

$$\begin{array}{ll} R \to P: & \{read\}_{ksenc},\ \mathsf{mac}(\{read\}_{ksenc}, kmac) \\ P \to R: & \{data_P\}_{ksenc},\ \mathsf{mac}(\{data_P\}_{ksenc}, kmac) \end{array}$$

where $data_P = \langle dg_P, \mathsf{sign}(\mathsf{hash}(dg_P), skDS), \mathsf{hash}(dg_P)\rangle$. This protocol is type-compliant, without requiring any additional tagging, using the same structure-preserving typing system as before. Type-compliance arises here from the fact that encrypted subterms in the formal specification of the protocol use a fixed key, the session key shared between the reader and the passport, preventing unification between messages other than the honest execution of the protocol.

Active authentication. This protocol can be informally described as follows.

$$\begin{array}{ll} R \to P: & \{init, r\}_{ksenc},\ \mathsf{mac}(\{init, r\}_{ksenc}, kmac) \\ P \to R: & \{\mathsf{sign}(\langle n, r\rangle, skP)\}_{ksenc},\ \mathsf{mac}(\{\mathsf{sign}(\langle n, r\rangle, skP)\}_{ksenc}, kmac) \end{array}$$

This protocol is type-compliant, without requiring any additional tagging, using the same structure-preserving typing system as before, and following the same argument as for the PA protocol.

Unfortunately, the BAC protocol is out of scope of our result since our syntax does not allow one to model else branches.

7.4 E-voting protocols

Helios [3] is an e-voting protocol that has been used in several real-life elections. We consider here a simplified version of this protocol (e.g. the authentication mechanism is replaced by a digital signature) whose informal description is given below:

$$V_i \to S : \quad \mathsf{sign}(\mathsf{raenc}(v_i, \mathsf{pub}(sks), r_i), ski)$$
$$S \to V_1, \ldots, V_n : \quad v_{i_1}, \ldots, v_{i_n}$$

First, the voter V_i casts her vote $v_i \in \{\mathsf{yes}, \mathsf{no}\}$. This is achieved by sending to the server a randomized encryption of v_i with the public key $\mathsf{pub}(sks)$ of the server S. Then, after receiving the ballot, the server outputs the tally. The server simply sends the valid votes received during the voting phase, in some random order.

As explained in Section 2.4, vote privacy can be modeled through the following equivalence

$$(\{P_S, P_V(c_a, ska, \mathsf{yes}, r_a), P_V(c_b, skb, \mathsf{no}, r_b)\}; \phi_0; \emptyset; 0)$$
$$\approx_t$$
$$(\{P_S, P_V(c_a, ska, \mathsf{no}, r_a), P_V(c_b, skb, \mathsf{yes}, r_b)\}; \phi_0; \emptyset; 0)$$

This corresponds to a simple scenario with two honest voters, where

- $P_S = \mathsf{in}(c, \mathsf{sign}(\mathsf{raenc}(x_1, \mathsf{pub}(sks), z_1), ska)).$
  $\mathsf{in}(c, \mathsf{sign}(\mathsf{raenc}(x_2, \mathsf{pub}(sks), z_2), skb)).(\mathsf{out}(c, x_1) \mid \mathsf{out}(c, x_2))$
- $P_V(c, sk, v, r) = \mathsf{out}(c, \mathsf{sign}(\mathsf{raenc}(v, \mathsf{pub}(sks), r), sk));$
- $\phi_0 = \{\mathsf{w}_1 \triangleright \mathsf{pub}(sks), \mathsf{w}_2 \triangleright \mathsf{vk}(ska), \mathsf{w}_3 \triangleright \mathsf{vk}(skb)\}.$

Type-compliance w.r.t. to a structure-preserving typing system can be obtained by simply giving the same initial type to r_a, r_b, z_1, and z_2, and we have also to give the same type to yes, no, x_1, and x_2. Thus, our typing result applies. In particular, if there is a witness of non equivalence, then there is a well-typed witness of non equivalence. Note that our protocol is not determinate due to the two outputs that occur in parallel on channel c. This means that our typing result holds for our notion of equivalence $\approx_t$ only (with static inclusion), we cannot conclude for trace equivalence with static equivalence.

We may want to consider a richer scenario involving a dishonest voter skc. The modelling will be similar. We need to modify the server role to allow 3 ballots to be received. The dishonest voter does not need to be modeled. It is sufficient to add skc to the knowledge of the attacker (i.e. in ϕ_0). Type-compliance holds with a similar typing system. Actually, this simplified version of the Helios protocol does not guarantee vote privacy (hence the corresponding equivalence does not hold). This is due to the fact that we do not model that the server is supposed to discard duplicated ciphertexts. The dishonest voter with key skc can intercept the ciphertext of voter A, sign it, and send it to the server. The result of the election will be different on both sides of the equivalence, and this is observable by the attacker. The corresponds to the replay attack described in [30].

To model a server performing such a weeding phase (where duplicated ballots are discarded), a non trivial else branch is needed. This is out of scope of our result.

8 CONCLUSION

We have established a simplification result for both reachability and equivalence properties: if there is an attack, then there is a well-typed attack, which reduces the search space. Our result holds for a large class of cryptographic primitives, that encompasses asymmetric encryption, signatures, and hashes, and for a flexible notion of typing system, provided the underlying protocol is type-compliant: any two unifiable encrypted subterms have the same type. Note that instead of considering the usual notion of trace equivalence, we study a weaker equivalence, which appears to coincide with the original one for determinate processes. As furture work, we plan to explore how to lift our simplification result to standard trace equivalence. The difficulty is that it requires to transfer equalities both ways (from P to Q and conversely) for each trace inclusion. In our proof, those equalities are preserved through type compliance on the left side. Thus, to handle standard trace equivalence, we would first need to relate the typing systems of both protocols under study.

Another natural next step would be to extend results that rely on the previous small attack result [23], limited to symmetric encryption. For example, we could probably establish a novel decidability result for protocols with nonces, for an unbounded number of sessions, for a large class of cryptographic primitives, generalizing the approach of [24]. Similarly, we plan to extend the tool SAT-Equiv [28] that (efficiently) decides trace equivalence for a bounded number of sessions. SAT-Equiv is currently limited to symmetric encryption as it relies on [23]. The extension to asymmetric encryption, signatures and hashes will require further (provably correct) optimizations, in order to preserve the efficiency of the tool.

Finally, typing results can potentially be used for composition results. Given two secure protocols P and Q, they can be safely composed if, intuitively, the execution of P does not interfere with the execution of Q (and conversely). Such composition results have been established for trace properties [4, 25, 29] as well as equivalence properties [5]. We could probably use our typing result to establish composition for equivalence properties and a larger class of primitives and protocols, assuming that protocols have disjoint types, which can be typically enforced by tagging.

Acknowledgments

This work has been partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreements No 645865-SPOOC and No 714955-POPSTAR, the ANR project TECAP No ANR-17-CE39-0004-01, and the DGA.

REFERENCES

[1] M. Abadi and C. Fournet. 2001. Mobile Values, New Names, and Secure Communication. In *Proc. 28th Symposium on Principles of Programming Languages (POPL'01)*. ACM Press.

[2] M. Abadi and R. M. Needham. 1996. Prudent Engineering Practice for Cryptographic Protocols. *IEEE Trans. Software Eng.* 22, 1 (1996), 6–15.

[3] Ben Adida. 2008. Helios: Web-based Open-Audit Voting. In *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*. USENIX Association, 335–348.

[4] O. Almousa, S. Mödersheim, P. Modesti, and L. Viganò. 2015. Typing and Compositionality for Security Protocols: A Generalization to the Geometric Fragment. In *Proc. 20th European Symposium on Research in Computer Security (ESORICS'15)*.

[5] M. Arapinis, V. Cheval, and S. Delaune. 2015. Composing security protocols: from confidentiality to privacy. In *Proc. 4th International Conference on Principles of Security and Trust (POST'15) (Lecture Notes in Computer Science)*, Vol. 9036. Springer, London, UK, 324–343.

[6] M. Arapinis, T. Chothia, E. Ritter, and M. Ryan. 2010. Analysing Unlinkability and Anonymity Using the Applied Pi Calculus. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*. IEEE Computer Society Press, 107–121.

[7] M. Arapinis and M. Duflot. 2007. Bounding messages for free in security protocols. In *Proc. 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*.

[8] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. 2005. The AVISPA Tool for the automated validation of Internet security protocols and applications. In *Proc. 17th International Conference on Computer Aided Verification, (CAV'2005) (LNCS)*, Vol. 3576. 281–285.

[9] M. Backes, C. Hritcu, and M. Maffei. 2008. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-Calculus. In *Proc. 21st IEEE Computer Security Foundations Symposium (CSF'08)*. IEEE Computer Society.

[10] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. 2018. A Formal Analysis of 5G Authentication. In *ACM CCS 2018 - 25th ACM Conference on Computer and Communications Security*, Vol. 14. ACM Press, Toronto, Canada.

[11] M. Baudet. 2005. Deciding Security of Protocols against Off-line Guessing Attacks. In *12th ACM Conference on Computer and Communications Security (CCS'05)*. ACM Press.

[12] B. Blanchet. 2001. An efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press.

[13] B. Blanchet. 2008. *Vérification automatique de protocoles cryptographiques : modèle formel et modèle calculatoire. Automatic verification of security protocols: formal model and computational model.* Mémoire d'habilitation à diriger des

recherches. Université Paris-Dauphine.
[14] B. Blanchet, M. Abadi, and C. Fournet. 2008. Automated Verification of Selected Equivalences for Security Protocols. *Journal of Logic and Algebraic Programming* 75, 1 (2008), 3–51.
[15] B. Blanchet and A. Podelski. 2003. Verification of Cryptographic Protocols: Tagging Enforces Termination. In *Foundations of Software Science and Computation Structures (FoSSaCS'03)*.
[16] Bruno Blanchet and Ben Smyth. 2018. Automated reasoning for equivalences in the applied pi calculus with barriers. *Journal of Computer Security* 26, 3 (2018), 367–422.
[17] M. Bruso, K. Chatzikokolakis, and J. den Hartog. 2010. Formal verification of privacy for RFID systems. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*.
[18] Mayla Brusó, Konstantinos Chatzikokolakis, Sandro Etalle, and Jerry Den Hartog. 2012. Linking Unlinkability. In *TGC 2012 - 7th International Symposium on Trustworthy Global Computing (TGC 2012: Trustworthy Global Computing)*, Vol. 8191. Springer, Newcastle upon Tyne, United Kingdom, 129–144. https://doi.org/10.1007/978-3-642-41157-1_9
[19] R. Chadha, Ş. Ciobâcă, and S. Kremer. 2012. Automated verification of equivalence properties of cryptographic protocols. In *Proc. 21th European Symposium on Programming (ESOP'12) (LNCS)*.
[20] V. Cheval, H. Comon-Lundh, and S. Delaune. 2011. Trace Equivalence Decision: Negative Tests and Non-determinism. In *PRoc. 18th ACM Conference on Computer and Communications Security (CCS'11)*. ACM.
[21] V. Cheval, V. Cortier, and S. Delaune. 2013. Deciding equivalence-based properties using constraint solving. *Theoretical Computer Science* 492 (June 2013), 1–39.
[22] R. Chrétien, V. Cortier, and S. Delaune. 2013. From security protocols to pushdown automata. In *Proc. 40th International Colloquium on Automata, Languages and Programming (ICALP'13)*.
[23] R. Chrétien, V. Cortier, and S. Delaune. 2014. Typing messages for free in security protocols: the case of equivalence properties. In *Proc. of the 25th International Conference on Concurrency Theory (CONCUR'14) (Lecture Notes in Computer Science)*. Springer, Rome, Italy.
[24] R. Chrétien, V. Cortier, and S. Delaune. 2015. Decidability of trace equivalence for protocols with nonces. In *Proc. 28th IEEE Computer Security Foundations Symposium (CSF'15)*. IEEE Computer Society Press.
[25] Ş Ciobâcă and V. Cortier. 2010. Protocol composition for arbitrary primitives. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF'10)*. IEEE Computer Society Press, Edinburgh, Scotland, UK, 322–336.
[26] H. Comon-Lundh and V. Cortier. 2003. New Decidability Results for Fragments of First-Order Logic and Application to Cryptographic Protocols. In *Proc. 14th International Conference on Rewriting Techniques and Applications (RTA'2003) (LNCS)*, Vol. 2706. Springer.
[27] H. Comon-Lundh, V. Cortier, and E. Zalinescu. 2010. Deciding security properties for cryptographic protocols. Application to key cycles. *ACM Transactions on Computational Logic (TOCL)* 11, 4 (2010).
[28] V. Cortier, A. Dallon, and S. Delaune. 2017. SAT-Equiv: an efficient tool for equivalence properties. In *Proc. 30th IEEE Computer Security Foundations Symposium (CSF'17)*. IEEE Computer Society Press, Santa Barbara, CA, USA.
[29] V. Cortier and S. Delaune. 2009. Safely Composing Security Protocols. *Formal Methods in System Design* 34, 1 (Feb. 2009), 1–36.
[30] Véronique Cortier and Ben Smyth. 2013. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security* 21, 1 (2013), 89–148.
[31] C. Cremers. 2008. The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In *Computer Aided Verification (CAV'08) (LNCS)*, Vol. 5123/2008. Springer, 414–418.
[32] S. Delaune, S. Kremer, and M. D. Ryan. 2008. Verifying Privacy-type Properties of Electronic Voting Protocols. *Journal of Computer Security* 4 (July 2008), 435–487.
[33] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. 1999. Undecidability of bounded security protocols. In *Workshop on Formal Methods and Security Protocols*. Trento, Italia.
[34] J. D. Guttman and F. Javier Thayer. 2000. Protocol Independence through Disjoint Encryption.. In *Proc. 13th Computer Security Foundations Workshop (CSFW'00)*. IEEE Comp. Soc. Press.
[35] J. Heather, G. Lowe, and S. Schneider. 2003. How to prevent type flaw attacks on security protocols. *Journal of Computer Security* 11, 2 (2003), 217–244.
[36] A. V. Hess and S. Mödersheim. 2017. Formalizing and Proving a Typing Result for Security Protocols in Isabelle/HOL. In *Proc. 30th IEEE Computer Security Foundations Symposium (CSF'17)*.
[37] G. Lowe. 1996. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96) (LNCS)*, Vol. 1055. Springer-Verlag, 147–166.
[38] G. Lowe. 1998. Towards a Completeness Result for Model Checking of Security Protocols. In *Proc. of the 11th Computer Security Foundations Workshop (CSFW'98)*. IEEE Computer Society Press.
[39] J. Millen and V. Shmatikov. 2001. Constraint Solving for Bounded-Process Cryptographic Protocol Analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*. ACM Press.

[40] R. Ramanujam and S. P. Suresh. 2003. Tagging Makes Secrecy Decidable with Unbounded Nonces as Well. In *Proc. 3rd Conference of Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03) (LNCS)*. Springer, 363–374.

[41] M. Rusinowitch and M. Turuani. 2003. Protocol Insecurity with Finite Number of Sessions and Composed Keys is NP-complete. *Theoretical Computer Science* 299 (April 2003), 451–475.

[42] B. Schmidt, S. Meier, C. Cremers, and D. Basin. 2012. Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In *Proc. 25th IEEE Computer Security Foundations Symposium (CSF'12)*. 78–94.

[43] A. Tiu and J. E. Dawson. 2010. Automating Open Bisimulation Checking for the Spi Calculus. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF'10)*. 307–321.

A PROOFS OF SECTION 4

A.1 Some preliminaries

Lemma 4.3. *Let ϕ be a Σ_0-frame, R a Σ_0-recipe such that $R\phi\downarrow$ is a Σ_0-message, and R' be such that $R \twoheadrightarrow R'$. We have that R' is a Σ_0-recipe, and $R'\phi\downarrow = R\phi\downarrow$.*

Proof. Since $R \twoheadrightarrow R'$, we know that there exists a position p in R, a rewriting rule $\ell' \twoheadrightarrow r$ associated to $\ell \to r \in \mathcal{R}$, and a substitution σ such that $R|_p = \ell'\sigma$, and $R' = R[r\sigma]_p$. Since $R\phi\downarrow$ is a Σ_0-message, we know that $(\ell'\sigma)\phi\downarrow$ is a Σ_0-message. Let $t'_1, \ldots, t'_k$ and $\mathsf{des} \in \Sigma_\mathsf{d}$ be such that $\ell' = \mathsf{des}(t'_1, \ldots, t'_n)$. We have that $(\ell'\sigma)\phi = \mathsf{des}((t'_1\sigma)\phi, \ldots, (t'_k\sigma)\phi)$. As $(\ell'\sigma)\phi\downarrow$ is a Σ_0-message, it means that $\ell \to r$ applies as it is the only rule reducing des. Hence, we have that $\mathsf{des}((t'_1\sigma)\phi\downarrow, \ldots, (t'_k\sigma)\phi\downarrow) \to (r\sigma)\phi\downarrow$. We deduce that $(\ell'\sigma)\phi\downarrow = (r\sigma)\phi\downarrow$, and therefore we have that $R\phi\downarrow = R'\phi\downarrow$. □

Lemma 4.5. *Let θ be a substitution with $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$ and whose image contains Σ_0^--recipes. Let R be a Σ_0^+-recipe in normal form w.r.t. $\twoheadrightarrow$ such that $(R\theta)\phi\downarrow$ is a Σ_0^+-message for some Σ_0^--frame ϕ. We have that R' is a simple Σ_0^+-recipe for any $R' \in St(R)$.*

Proof. We prove this result by structural induction on R.
Base case: $R \in \mathcal{W} \cup \Sigma_0^+$. In both cases, the result holds since R is a simple Σ_0^+-recipe.
Induction case: We have that $R = \mathsf{f}(R_1, \ldots, R_k)$ for some $\mathsf{f} \in \Sigma$, and we know that $R_1, \ldots, R_k$ are in normal form w.r.t. $\downarrow$.

- *Case* $\mathsf{f} \in \Sigma_\mathsf{c}$. We have that $(R\theta)\phi\downarrow = \mathsf{f}((R_1\theta)\phi\downarrow, \ldots, (R_k\theta)\phi\downarrow)$ is a Σ_0^+-message for some Σ_0^--frame ϕ, and thus $(R_i\theta)\phi\downarrow$ is a Σ_0^+-message for $i \in \{1, \ldots, k\}$. Applying our induction hypothesis on R_i ($1 \le i \le k$), we easily conclude.
- *Case* $\mathsf{f} = \mathsf{des} \in \Sigma_\mathsf{d}$. Let $\ell_{\mathsf{des}} \to r_{\mathsf{des}}$ be the rule in $\mathcal{R}$ such that $\mathsf{root}(\ell_{\mathsf{des}}) = \mathsf{des}$, and $\ell'_{\mathsf{des}} \twoheadrightarrow r_{\mathsf{des}}$ its associated forced rewriting rule. If $r_{\mathsf{des}} \in \mathcal{T}_0(\Sigma_\mathsf{c}, \emptyset)$ then we have that $R \twoheadrightarrow r_{\mathsf{des}}$, which is impossible as R is in normal form w.r.t. $\twoheadrightarrow$. Thus, there is a unique position p_0 of ℓ_{des} such that $\ell_{\mathsf{des}}|_{p_0} = r_{\mathsf{des}}$ and $p_0 = 1.p'_0$. We know that each R_i is in normal form w.r.t. $\twoheadrightarrow$, and we have $(R_i\theta)\phi\downarrow$ is a Σ_0^+-message ($1 \le i \le k$) as $(R\theta)\phi\downarrow$ is a Σ_0^+-message. Thus, our induction hypothesis applies and we deduce that any subterm of R_i is a simple Σ_0^+-recipe ($1 \le i \le k$).
  We first assume that R_1 is a subterm Σ_0^+-recipe. Let ψ be Σ_0^+-frame such that $R\psi\downarrow$ is a Σ_0^+-message. We have that $R\psi\downarrow \in St(R_1\psi\downarrow) \subseteq St(\psi)$. Thus, R is a subterm Σ_0^+-recipe, and thus a simple Σ_0^+-recipe.
  Otherwise, we have that $R_1 = C[R'_1, \ldots, R'_{k'}]$ for some context built using symbols from $\Sigma_\mathsf{c} \uplus \Sigma_0^+$, and each R'_j ($1 \le j \le k'$) is a subterm Σ_0^+-recipe such that $\mathsf{root}(R'_j) \notin \Sigma_\mathsf{c}$. We have that $R = \mathsf{des}(C[R'_1, \ldots, R'_{k'}], R_2, \ldots, R_k)$, and p_0 does not correspond to a position of the context C since R is in normal form w.r.t. $\twoheadrightarrow$. Let p' be the longest prefix of p'_0 that corresponds to a position of C. We have that $C[R'_1, \ldots, R'_{k'}]|_{p'} = R'_j$ for some j. Let ψ be Σ_0^+-frame such that $R\psi\downarrow$ is a Σ_0^+-message. We have that $R\psi\downarrow \in St(R'_j\psi\downarrow) \subseteq St(\psi)$ as R'_j is a subterm Σ_0^+-recipe. Thus, R is a subterm Σ_0^+-recipe, and thus a simple Σ_0^+-recipe.

This allows us to conclude. □

A.2 Our measure

Lemma A.1. *Let ϕ_S be a Σ_0^+-frame, and R be a Σ_0^+-recipe. We have that $\mu^2_{\phi_S}(R) \ge 0$.*

Proof. Let $R = R_0[R_1, \ldots, R_n]$ where R_0 is the hat of R. Since R_0 only contains constructors, we have that $R\phi_S\downarrow = R_0[R_1\phi_S\downarrow, \ldots, R_n\phi_S\downarrow]$. Hence, we have that $|R\phi_S\downarrow| \ge |R_0| = \mathsf{hat}(R)$, which implies $\mu^2_{\phi_S}(R) \ge 0$. □

Lemma A.2. *Let $\prec$ be an ordering on Σ_{fresh}, $\ell_{\mathsf{des}} \to r_{\mathsf{des}}$ be a rewriting rule from $\mathcal{R}$ (as defined in Section 2.1), and σ be a substitution such that $img(\sigma) \subseteq \mathcal{T}(\Sigma, \Sigma_0^+ \uplus \mathcal{N})$.*

We have that $\mathsf{Multi}(r_{\mathsf{des}}\sigma) < \mathsf{Multi}(\ell_{\mathsf{des}}\sigma)$. This result also holds when considering the forced rewriting associated to a rewriting rule in $\mathcal{R}$.

Proof. First, we consider the case where $r_{\mathsf{des}} \in \mathcal{T}_0(\Sigma_c, \emptyset)$. In such a case, we have

$$\mathsf{Multi}(r_{\mathsf{des}}\sigma) = \mathsf{Multi}(r_{\mathsf{des}}) < \{\mathsf{des}\} < \mathsf{Multi}(\ell_{\mathsf{des}}\sigma)$$

Now, we consider the case where $r_{\mathsf{des}} \in St(\ell_{\mathsf{des}})$. We have that:

$$\mathsf{Multi}(r_{\mathsf{des}}\sigma) < \{\mathsf{des}\} \uplus \mathsf{Multi}(r_{\mathsf{des}}\sigma) \leq \mathsf{Multi}(\ell_{\mathsf{des}}\sigma).$$

Thus, in both cases, we have that $\mathsf{Multi}(r_{\mathsf{des}}\sigma) < \mathsf{Multi}(\ell_{\mathsf{des}}\sigma)$. The proof regarding the case of a forced rewriting rule can be done in similar way. □

Lemma A.3. *Let $\prec$ be an ordering on Σ_{fresh}, $\mathsf{f} \in \Sigma$ of arity k, and $t_1, \ldots, t_k \in \mathcal{T}(\Sigma, D)$ for some set D of data. We have that $\mathsf{Multi}(\mathsf{f}(t_1, \ldots, t_k)\downarrow) \leq \mathsf{Multi}(\mathsf{f}(t_1\downarrow, \ldots, t_k\downarrow))$, and similarly for $\downarrow$.*

Proof. First, we consider the case where $\mathsf{f}(t_1, \ldots, t_k)\downarrow = \mathsf{f}(t_1\downarrow, \ldots, t_k\downarrow)$. In such a situation, the result trivially holds. Thus, we know that $\mathsf{f} = \mathsf{des} \in \Sigma_{\mathsf{d}}$, and $\mathsf{des}(t_1, \ldots, t_k)\downarrow \neq \mathsf{des}(t_1\downarrow, \ldots, t_k\downarrow)$. It means that there exists a substitution σ such that:

$$\mathsf{des}(t_1\downarrow, \ldots, t_k\downarrow) = \ell_{\mathsf{des}}\sigma \text{ and } \mathsf{des}(t_1, \ldots, t_k)\downarrow = r_{\mathsf{des}}\sigma.$$

Thanks to Lemma A.2, we know that $\mathsf{Multi}(\mathsf{des}(t_1, \ldots, t_k)\downarrow < \mathsf{Multi}(\mathsf{des}(t_1\downarrow, \ldots, t_k\downarrow))$. This concludes the proof. A similar reasoning allows us to conclude regarding $\Downarrow$. □

Lemma 4.6. *Let $\prec$ be an ordering on Σ_{fresh}, ϕ_S be a Σ_0^+-frame, R_1, R_2 be two Σ_0^+-recipes such that $R_2\phi_S\downarrow$ is not a Σ_0^+-message, and $\mu^1_{\phi_S}(R_1) < \mu^1_{\phi_S}(R_2)$. Let R_0 be a Σ_0^+-recipe, and p a position in R_0. We have that:*

$$\mu^1_{\phi_S}(R_0[R_1]_p) < \mu^1_{\phi_S}(R_0[R_2]_p).$$

Proof. We first establish the following claim by induction on the length of p ($\Box$ is not a message).

$$\mathsf{Multi}(R_0[R_1]_p\phi_S\downarrow) \leq (\mathsf{Multi}(R_0\phi_S[\Box]_p\downarrow) \setminus \{\Box\}) \uplus \mathsf{Multi}(R_1\phi_S\downarrow).$$

Base case: $p = \epsilon$. We have $\mathsf{Multi}(R_0\phi_S[\Box]_p\downarrow) = \{\Box\}$. So

$$\mathsf{Multi}(R_0[R_1]_p\phi_S\downarrow) = \mathsf{Multi}(R_1\phi_S\downarrow) = (\mathsf{Multi}(R_0\phi_S[\Box]_p\downarrow) \setminus \{\Box\}) \uplus \mathsf{Multi}(R_1\phi_S\downarrow)$$

Inductive case: $p = j.p'$, and $R_0 = \mathsf{f}(R'_1, \ldots, R'_k)$ for some Σ_0^+-recipes $R'_1, \ldots, R'_k$. Thanks to Lemma A.3, we obtain $\mathsf{Multi}(R_0[R_1]_p\phi_S\downarrow) \leq \mathsf{Multi}(\mathsf{f}(R'_1\phi_S\downarrow, \ldots, R'_j[R_1]_{p'}\phi_S\downarrow, \ldots, R'_k\phi_S\downarrow))$, and thus

$$\mathsf{Multi}(R_0[R_1]_p\phi_S\downarrow) \leq \{\mathsf{f}\} \uplus (\uplus_{i\neq j}\mathsf{Multi}(R'_i\phi_S\downarrow)) \uplus \mathsf{Multi}(R'_j[R_1]_{p'}\phi_S\downarrow).$$

By induction hypothesis we have: $\mathsf{Multi}(R'_j[R_1\phi_S\downarrow]_{p'}) \leq (\mathsf{Multi}(R'_j\phi_S[\Box]_{p'}\downarrow) \setminus \{\Box\}) \uplus \mathsf{Multi}(R_1\phi_S\downarrow)$. Therefore, we have that:

$$\mathsf{Multi}(R_0[R_1]_p\phi_S\downarrow) \leq \{\mathsf{f}\} \uplus (\uplus_{i\neq j}\mathsf{Multi}(R'_i\phi_S\downarrow)) \uplus (\mathsf{Multi}(R'_j\phi_S[\Box]_{p'}\downarrow) \setminus \{\Box\}) \uplus \mathsf{Multi}(R_1\phi_S\downarrow)$$

But $R'_j\phi_S[\Box]_{p'}\downarrow$ is not a message, and thus no reduction can occur above $\Box$. Thus:

$$\mathsf{Multi}(R_0\phi_S[\Box]_p\downarrow) = \{\mathsf{f}\} \uplus (\uplus_{i\neq j}\mathsf{Multi}(R'_i\phi_S\downarrow)) \uplus \mathsf{Multi}(R'_j[\Box]_{p'}\phi_S\downarrow).$$

Hence, we deduce that:

$$\mathsf{Multi}(R_0\phi_S[\Box]_p\downarrow) \setminus \{\Box\} = \{\mathsf{f}\} \uplus (\uplus_{i\neq j}\mathsf{Multi}(R'_i\phi_S\downarrow) \uplus (\mathsf{Multi}(R'_j[\Box]_{p'}\phi_S\downarrow) \setminus \{\Box\})$$

We deduce:

$$\mathsf{Multi}(R_0[R_1]_p\phi_S{\downarrow}) \leq (\mathsf{Multi}(R_0[\Box]_p\phi_S{\downarrow}) \setminus \{\Box\}) \uplus \mathsf{Multi}(R_1\phi_S{\downarrow})$$

This proves the claim.

Now, as $R_2\phi_S{\downarrow}$ is not a message, $R_0[R_2]_p\phi_S{\downarrow} = (\mathsf{Multi}(R_0\phi_S[\Box]_p{\downarrow}) \setminus \{\Box\}) \uplus \mathsf{Multi}(R_2\phi_S{\downarrow})$. Since $\mu^1_{\phi_S}(R_1) < \mu^1_{\phi_S}(R_2)$, relying on our claim, we easily deduce that $\mu^1_{\phi_S}(R_0[R_1]_p) < \mu^1_{\phi_S}(R_0[R_2]_p)$. □

Lemma 4.7. *Let $\prec$ be an ordering on Σ_{fresh}, ϕ_S be a Σ_0^+-frame, and R, R' be two Σ_0^+-recipes such that $R \twoheadrightarrow R'$. We have that $\mu_{\phi_S}(R') < \mu_{\phi_S}(R)$.*

Proof. We first consider the case where $R\phi_S{\downarrow} = R'\phi_S{\downarrow}$. Hence, we have that $\mu^1_{\phi_S}(R) = \mu^1_{\phi_S}(R')$, and we have also that $\mu^3(R) < \mu^3(R')$. Since $|R\phi_S{\downarrow}| = |R'\phi_S{\downarrow}|$, in order to conclude, it only remains to establish that $|\mathsf{hat}(R')| \geq |\mathsf{hat}(R)|$. We have that $R = R_0[R_1, \ldots, R_n]$ with $R_0 = \mathsf{hat}(R)$. Since $R \twoheadrightarrow R'$, we know that $R' = R_0[R_1, \ldots, R'_i, \ldots, R_n]$ with $R_i \twoheadrightarrow R'_i$, and thus $|\mathsf{hat}(R')| \geq |R_0|$.

Now, we consider the case where $R\phi_S{\downarrow} \neq R'\phi_S{\downarrow}$. Let $\ell'_{\mathsf{des}} \twoheadrightarrow r_{\mathsf{des}}$ be the rule applied to rewrite R in R'. We have that $R = R[\ell'_{\mathsf{des}}\delta]_p$ and $R' = R[r_{\mathsf{des}}\delta]_p$ for some position p, and some substitution δ. Therefore, we have that $R\phi_S{\downarrow} = (R\phi_S)[(\ell'_{\mathsf{des}}\delta)\phi_S{\downarrow}]_p{\downarrow}$ and $R'\phi_S{\downarrow} = (R\phi_S)[(r_{\mathsf{des}}\delta)\phi_S{\downarrow}]_p{\downarrow}$. By Lemma A.2, and as $([\ell'_{\mathsf{des}}\delta]\phi_S{\downarrow})$ does not reduce (otherwise we would have that $R\phi_S{\downarrow} = R'\phi_S{\downarrow}$) we have that $\mathsf{Multi}((\ell'_{\mathsf{des}}\delta)\phi_S{\downarrow}) > \mathsf{Multi}([r_{\mathsf{des}}\delta]\phi_S{\downarrow})$. We deduce that $\mu^1_{\phi_S}(\ell'_{\mathsf{des}}\delta) > \mu^1_{\phi_S}(r_{\mathsf{des}}\delta)$. Then Lemma 4.6 allows to conclude that $\mu^1_{\phi_S}(R') < \mu^1_{\phi_S}(R)$. □

A.3 Completeness

Lemma A.4. *Let ϕ_S be a Σ_0^+-frame together with $\prec$ a total ordering on $dom(\phi_S)$. Let θ be a substitution such that $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$, and for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in ϕ_S we have that $c \in dom(\theta)$ and $c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S))$. Moreover, we assume that in case $c \in \Sigma_{\mathsf{fresh}}$ occurs in $\mathsf{w}\phi_S$, then $\mathsf{w}' \prec \mathsf{w}$ for any $\mathsf{w}' \in vars(c\theta)$.*

We have that any constant $c' \in \Sigma_{\mathsf{fresh}}$ that occurs in $(c\theta)\phi_S{\downarrow}$ is such that $\mathsf{rank}(c') \prec \mathsf{rank}(c)$ where $\mathsf{rank}(c) = \max_{\prec}\{\mathsf{w} \mid \mathsf{w} \in vars(c\theta)\})$ when $\{\mathsf{w} \in vars(c\theta)\} \neq \emptyset$, and $\bot$ otherwise.

Proof. Let ϕ_S be a Σ_0^+-frame as defined in the lemma, and $c \in dom(\theta)$. Let $c' \in \Sigma_{\mathsf{fresh}}$ be a constant that occurs in $(c\theta)\phi_S{\downarrow}$. In case $\mathsf{rank}(c) = \bot$, then it means that $vars(c\theta) = \emptyset$, and thus no constant from Σ_{fresh} occurs in $c\theta$. Therefore, we are done. Now, let $\mathsf{w}_i = \mathsf{rank}(c)$, we have that $vars(c\theta) \subseteq \{\mathsf{w} \mid \mathsf{w} \preceq \mathsf{w}_i\}$, and by hypothesis on θ, we have that $\mathsf{rank}(c') \prec \mathsf{w}_i$ for any constant $c' \in \Sigma_{\mathsf{fresh}}$ occurring in $(c\theta)\phi_S$, and thus we have that $\mathsf{rank}(c') \prec \mathsf{rank}(c)$. □

Lemma 4.8. *Let ϕ_S be a Σ_0^+-frame together with $\prec$ a total ordering on $dom(\phi_S)$. Let θ be a substitution such that $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$, and for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in ϕ_S we have that $c \in dom(\theta)$ and $c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S))$. Moreover, we assume that in case $c \in \Sigma_{\mathsf{fresh}}$ occurs in $\mathsf{w}\phi_S$, then $\mathsf{w}' \prec \mathsf{w}$ for any $\mathsf{w}' \in vars(c\theta)$. We consider the substitution λ whose domain is $dom(\theta)$, and such that:*

$$c\lambda = (c\theta)(\phi_S\lambda){\downarrow} \text{ for any } c \in dom(\lambda).$$

The substitution λ is well-defined. Moreover, if $(c\theta)\phi_S{\downarrow}$ is a Σ_0^+-message for each $c \in dom(\theta)$, and $(c\theta)\phi_S{\downarrow}$ is an atomic Σ_0^+-message when $c \in \Sigma^{\mathsf{atom}}_{\mathsf{fresh}}$, then $c\lambda$ is a Σ_0^--message for each $c \in dom(\lambda)$, and $c\lambda$ is an atomic Σ_0^--message when $c \in \Sigma^{\mathsf{atom}}_{\mathsf{fresh}}$.

We call λ the first-order substitution associated to θ through ϕ_S.

Proof. Let $dom(\phi_S) = \{\mathsf{w}_1, \ldots, \mathsf{w}_n\}$ be such that $\mathsf{w}_1 \prec \mathsf{w}_2 \prec \ldots \prec \mathsf{w}_n$. We show this result by induction on $\mathsf{rank}(c)$ relying on the order $\prec$.

Base case: $c \in dom(\theta)$ such that $\mathsf{rank}(c) = \bot$. In such a case, we have that $vars(c\theta) = \emptyset$, and thus $c\lambda = (c\theta){\downarrow}$ is well-defined. Moreover, assuming that $(c\theta)\phi_S{\downarrow} = (c\theta){\downarrow}$ is Σ_0^+-message, then we know

that it is actually a Σ_0^--message, and thus $c\lambda$ is a Σ_0^--message which is atomic in case $(c\theta)\phi_S\!\downarrow$ is atomic.

Inductive step: $c \in dom(\theta)$ *such that* $\mathsf{rank}(c) = \mathsf{w}_i$. Let λ_i be the substitution which coincides with λ on its domain $dom(\lambda_i) = \{c' \mid c' \in dom(\lambda) \text{ and } \mathsf{rank}(c') \prec \mathsf{w}_i\}$. We have that λ_i is well-defined, and actually we have that $w_j\phi_S\lambda_i = w_j\phi_S\lambda$ for any $\mathsf{w}_j \preceq \mathsf{w}_i$. Since $\mathsf{rank}(c) = \mathsf{w}_i$, we know that $vars(c\theta) \subseteq \{\mathsf{w}_1, \ldots, \mathsf{w}_i\}$, and thus $c\lambda = (c\theta)(\phi_S\lambda)\!\downarrow = (c\theta)(\phi_S\lambda_i)\!\downarrow$ is well-defined. Moreover, assuming that $(c\theta)\phi_S\!\downarrow$ is a Σ_0^+-message, then we know that any constant $c' \in \Sigma_{\mathsf{fresh}}$ occurring in $(c\theta)\phi_S\!\downarrow$ is such that $\mathsf{rank}(c') \prec \mathsf{w}_i$ (thanks to Lemma A.4), and thus $c'\lambda$ is a Σ_0^--message, and an atomic one when $c \in \Sigma^{\mathsf{atom}}_{\mathsf{fresh}}$. Thus, we have that $(c\theta)\phi_S\!\downarrow\lambda$ is a Σ_0^--message and an atomic one in case $c \in \Sigma^{\mathsf{fresh}}_{\mathsf{atom}}$. Actually, we have that $(c\theta)\phi_S\!\downarrow\lambda = (c\theta)(\phi_S\lambda)\!\downarrow$ which allows us to conclude. □

Lemma 4.9. *Let ϕ_S be a Σ_0^+-frame together with $\prec$ a total ordering on $dom(\phi_S)$. Let θ be a substitution such that $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$, and for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in ϕ_S we have that $c \in dom(\theta)$ and $c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S))$. Moreover, we assume that in case $c \in \Sigma_{\mathsf{fresh}}$ occurs in $\mathsf{w}\phi_S$, then $\mathsf{w}' \prec \mathsf{w}$ for any $\mathsf{w}' \in vars(c\theta)$. Let λ be the first-order substitution associated to θ through ϕ_S.*

Assume that for any $c \in dom(\lambda)$, we have that $c\lambda$ is a Σ_0^--message. Moreover, $c\lambda$ is an atomic Σ_0^--message when $c \in \Sigma^{\mathsf{atom}}_{\mathsf{fresh}}$. Let $R \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\theta) \uplus dom(\phi_S))$ such that $R\phi_S\!\downarrow$ is a Σ_0^+-message. We have that $(R\theta)(\phi_S\lambda)\!\downarrow = (R\phi_S\!\downarrow)\lambda$.

Proof. We prove this result by structural induction on R.
Base case: $R \in \Sigma_0^- \uplus dom(\theta) \uplus dom(\phi_S)$. In case $R = c_0 \in \Sigma_0^-$, then we have that $(c\theta)(\phi_S\lambda)\!\downarrow = c_0 = (c\phi_S\!\downarrow)\lambda)$. In case $R = c \in dom(\theta)$, then we have that $(c\theta)(\phi_S\lambda)\!\downarrow = c\lambda = (c\phi_S\!\downarrow)\lambda$ thanks to Lemma 4.8. In case $R = \mathsf{w} \in dom(\phi_S)$, then we have that $(c\theta)(\phi_S\lambda)\!\downarrow = \mathsf{w}\phi_S\lambda\!\downarrow = (\mathsf{w}\phi_S\!\downarrow)\lambda$.

Inductive step: $R\mathsf{f}(R_1, \ldots, R_k)$. In case $\mathsf{f} \in \Sigma_c$, then we have that:

$$(R\theta)(\phi_S\lambda)\!\downarrow = \mathsf{f}((R_1\theta)(\phi_S\lambda)\!\downarrow, \ldots (R_k\theta)(\phi_S\lambda)\!\downarrow) = \mathsf{f}(R_1\phi_S\!\downarrow\lambda, \ldots R_k\phi_S\!\downarrow\lambda) = R\phi_S\!\downarrow\lambda.$$

In case $\mathsf{f} \in \Sigma_{\mathsf{d}}$, then for each $i \in \{i, \ldots, k\}$, by hypothesis we have that $R_i\phi_S\!\downarrow$ is a Σ_0^+-message. Thus, relying on our induction hypothesis, we have that:

$$(R\theta)(\phi_S\lambda)\!\downarrow = \mathsf{f}((R_1\theta)(\phi_S\lambda)\!\downarrow, \ldots, (R_k\theta)(\phi_S\lambda)\!\downarrow)\!\downarrow = \mathsf{f}((R_1\phi_S)\!\downarrow\lambda, \ldots, (R_k\phi_S)\!\downarrow\lambda)\!\downarrow$$

However, we have that $R\phi_S\!\downarrow\lambda = \mathsf{f}(R_1\phi_S\!\downarrow, \ldots, R_k\phi_S\!\downarrow)\!\downarrow\lambda = \mathsf{f}(R_1\phi_S\!\downarrow, \ldots, R_k\phi_S\!\downarrow)\lambda\!\downarrow$ as $c\lambda$ is a Σ_0^--message whenever $c \in dom(\lambda)$ and $c\lambda$ is an atomic Σ_0^--message when $c \in \Sigma^{\mathsf{atom}}_{\mathsf{fresh}}$ by Lemma 4.8. So $R\phi_S\!\downarrow\lambda = \mathsf{f}(R_1\phi_S\!\downarrow\lambda, \ldots, R_k\phi_S\!\downarrow\lambda)\!\downarrow = (R\theta)(\phi_S\lambda)\!\downarrow$. □

Lemma 4.10. *Let $\mathcal{K}_0 = (\mathcal{P}_0; \phi_0; \emptyset; 0)$ be an initial Σ_0-configuration and $\mathcal{K} = (\mathcal{P}; \phi; \sigma; i)$ be a Σ_0-configuration such that $\mathcal{K}_0 \overset{\mathsf{tr}}{\Longrightarrow} \mathcal{K}$ for some* tr *w.r.t. Σ_0.*

(1) We have that $ESt(\mathcal{K}\sigma) \subseteq ESt(\mathcal{K}_0\sigma)$.
(2) Moreover, in case σ is an mgu between pairs of terms occurring in $ESt(\mathcal{K}_0)$, then we have that $ESt(\mathcal{K}\sigma) \subseteq ESt(\mathcal{K}_0)\sigma$.

Proof. We prove $ESt(\mathcal{K}\sigma') \subseteq ESt(\mathcal{K}_0\sigma')$ for any σ' which coincide on σ on $dom(\sigma)$ by induction on the execution $\mathcal{K}_0 \xrightarrow{\mathsf{tr}} \mathcal{K}$.

Base case: tr *is empty.* In such a case, the result is obvious.

Inductive case: $\mathsf{tr} = \mathsf{tr}_0 \cdot \alpha$. We have that $\mathcal{K}_0 \xrightarrow{\mathsf{tr}_0} \mathcal{K}_1 \xrightarrow{\alpha} \mathcal{K}$ where $\mathcal{K}_1 = (\mathcal{P}_1; \phi_1; \sigma_1; i_1)$, and $ESt(\mathcal{K}_1\sigma_1') \subseteq ESt(\mathcal{K}_0\sigma_1')$ for any σ_1' which coincide with σ_1 on $dom(\sigma_1)$. We distinguish three cases depending on α:

- In case α is either a τ action or a phase i action, then $\sigma = \sigma_1$ and $St(\mathcal{K}) = St(\mathcal{K}_1)$. Therefore, let σ' be a substitution which coincides with $\sigma = \sigma_1$ on $dom(\sigma)$, we have that $ESt(\mathcal{K}\sigma') = ESt(\mathcal{K}_1\sigma') \subseteq ESt(\mathcal{K}_0\sigma')$.
- In case α is an input, then $\mathcal{P}_1 = \mathsf{in}(c,u).P \uplus \mathcal{P}_1'$ and $\mathcal{P} = P \uplus \mathcal{P}_1'$ so $St(\mathcal{P}) \subseteq St(\mathcal{P}_1)$. Moreover, $\sigma = \sigma_1 \uplus \tau$ and $dom(\tau) = vars(u\sigma_1)$. Hence, we have that $ESt(\tau) \subseteq ESt(u\sigma_1\tau) = ESt(u\sigma)$. We also have $\phi = \phi_1$, and thus $ESt(\phi) = ESt(\phi_1)$. Therefore, Let σ' be a substitution which coincides with σ on $dom(\sigma)$. We have that $ESt(\mathcal{K}\sigma') \subseteq ESt(\mathcal{K}_1\sigma')$. Note that σ' coincides with σ_1 on $dom(\sigma_1)$, thus thanks to our induction hypothesis, we know that $ESt(\mathcal{K}_1\sigma') \subseteq ESt(\mathcal{K}_0\sigma')$.
- In case α is an output, then $\phi = \phi_1 \uplus \{\mathsf{w} \triangleright u\sigma_1\}$ for some $u \in St(\mathcal{K}_1)$ and $\sigma = \sigma_1$. Hence we have that $ESt(\phi) \subseteq ESt(\mathcal{K}_1\sigma_1)$. Let σ' be a substitution which coincides with $\sigma = \sigma_1$ on $dom(\sigma)$, we have that $ESt(\mathcal{K}\sigma') \subseteq ESt(\mathcal{K}_1\sigma') \subseteq ESt(\mathcal{K}_0\sigma')$.

This concludes the proof for the first item.

To prove item 2, we consider σ the mgu between pairs of terms occurring in $ESt(\mathcal{K}_0)$, and we show that $ESt(\mathcal{K}_0\sigma) \subseteq ESt(\mathcal{K}_0)\sigma$. Note that this inclusion together with item 1 allows us to conclude. Let $t \in ESt(\mathcal{K}_0\sigma)$ be such that $t \notin ESt(\mathcal{K}_0)\sigma$. Therefore, we have that $t \in ESt(img(\sigma))$. Let $\overline{\sigma} = \sigma\delta$ where δ replaces any occurrence of t in $img(\sigma)$ by a fresh variable x. We have that:

- $u\overline{\sigma} = v\overline{\sigma}$ for any $u = v \in \Gamma$. Indeed, we know that $u\sigma = v\sigma$, and thus $(u\sigma)\delta = (v\sigma)\delta$. Since $t \notin ESt(\mathcal{K}_0)\sigma$, and $u, v \in ESt(\mathcal{K}_0)$, we deduce that $(u\sigma)\delta = u(\sigma\delta) = u\overline{\sigma}$ and similarly $(v\sigma)\delta = v(\sigma\delta) = v\overline{\sigma}$.
- $\overline{\sigma}$ is strictly more general that σ. Indeed, we have that $\sigma = \overline{\sigma}\tau$ considering $\tau = \{x \mapsto t\}$ and t is an encrypted term, and thus not a variable.

This leads to a contradiction since $\sigma = mgu(\Gamma)$ and concludes the proof. □

Proposition 4.11. *Let $\mathcal{K}_P = (\mathcal{P}_0; \phi_0; \emptyset; i_0)$ be an initial Σ_0^--configuration, and $(\mathrm{tr}, \phi) \in \mathrm{trace}_{\Sigma_0^-}(\mathcal{K}_P)$ with underlying substitution σ. Then, there exists $(\mathrm{tr}_S, \phi_S) \in \mathrm{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ with underlying substitution σ_S with $dom(\sigma_S) = dom(\sigma)$ such that $\sigma_S = mgu(\Gamma)\rho$, as well as two substitutions λ and θ such that:*

- $\Gamma = \{(u,v) \mid u, v \in ESt(\mathcal{K}_P)$ *such that* $u\sigma = v\sigma\}$.
- *ρ is a bijective renaming from variables in $dom(\sigma) \setminus dom(mgu(\Gamma))$ to constants in Σ_{fresh} such that $x\rho \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$ if, and only if, $x\sigma$ is an atomic Σ_0^--message.*
- *$dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$, for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in tr_S, we have that $c \in dom(\theta)$, $c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S))$, and $\mathsf{w}' \prec c$ for any $\mathsf{w}' \in vars(c\theta)$ (where $\prec$ is the ordering induced by tr_S).*
- *for any $c \in dom(\theta)$, $(c\theta)\phi_S{\downarrow}$ is a Σ_0^+-message and it is an atom when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$.*
- *λ is the first-order substitution associated to θ through ϕ_S.*
- *$\phi = \phi_S\lambda$, $\sigma = \sigma_S\lambda$, and $(\mathrm{tr}_S\theta)\phi{\downarrow} = \mathrm{tr}\phi{\downarrow}$.*

We say that (ρ, λ, θ) is a well-typed concretization *of (tr_S, ϕ_S) w.r.t. (tr, ϕ) in the context $\mathcal{K}_P$ when ρ, λ, θ satisfy the 6 conditions listed above.*

Proof. We know that $\mathcal{K}_P \overset{\mathrm{tr}}{\Longrightarrow} (\mathcal{P}; \phi; \sigma; i)$. Let $\Gamma = \{(u,v) \mid u, v \in ESt(\mathcal{K}_P)$ such that $u\sigma = v\sigma\}$, and ρ be a bijective renaming from variables in $dom(\sigma) \setminus dom(mgu(\Gamma))$ to constants in Σ_{fresh} such that $x\rho \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$ if, and only if, $x\sigma$ is an atomic Σ_0^--message. Let σ_S be such that $dom(\sigma_S) = dom(\sigma)$ and $\sigma_S = mgu(\Gamma)\rho$. As $mgu(\Gamma)$ is more general than σ, we have $\sigma = mgu(\Gamma)\lambda_0$ for some λ_0. Let $\lambda = \rho^{-1}\lambda_0$. We have that $\sigma = \sigma_S\lambda$.

We show by induction on the length of a prefix $\mathcal{K}_P \overset{\mathrm{tr}^+}{\Longrightarrow} (\mathcal{P}^+; \phi^+; \sigma^+; i^+)$ of this execution trace that there exists $(\mathrm{tr}_S^+, \phi_S^+) \in \mathrm{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ with underlying substitution $\sigma_S^+ = \sigma_S|_{dom(\sigma^+)}$, as well as two substitutions θ^+ and λ^+ such that:

- $dom(\theta^+) \subseteq \Sigma_{\mathsf{fresh}}$, for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in tr_S^+, we have that $c \in dom(\theta^+)$, $c\theta^+ \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S^+))$, and $\mathsf{w}' \prec^+ c$ for any $\mathsf{w}' \in vars(c\theta^+)$ (where $\prec^+$ is the ordering induced by tr_S^+).
- for any $c \in dom(\theta^+)$, $(c\theta^+)\phi_S^+\!\downarrow$ is a Σ_0^+-message and it is an atom when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$.
- λ^+ is the first-order substitution associated to θ^+ through ϕ_S^+.
- $\phi^+ = \phi_S^+\lambda^+$, $\sigma^+ = \sigma_S^+\lambda^+$, and $(\mathsf{tr}_S^+\theta^+)\phi^+\!\downarrow = \mathsf{tr}^+\phi^+\!\downarrow$.

Base case: tr^+ is empty. In such a case, we have that $dom(\sigma^+) = \emptyset$, and $\phi^+ = \phi_0$. Let $\mathsf{tr}_S^+ = \epsilon$, $\phi_S^+ = \phi_0$, and $\sigma_S^+ = \emptyset$. Choosing θ^+ and λ^+ such that $dom(\theta^+) = dom(\lambda^+) = \emptyset$, the result trivially holds.

Inductive case: $\mathsf{tr}^+ = \mathsf{tr}^-.\alpha$. In such a case, we have that:

$$\mathcal{K}_P = (\mathcal{P}_0; \phi_0; \emptyset; i_0) \xRightarrow{\mathsf{tr}^-} (\mathcal{P}^-; \phi^-; \sigma^-; i^-) \xRightarrow{\alpha} (\mathcal{P}^+; \phi^+; \sigma^+; i^+).$$

Thanks to our induction hypothesis applied on tr^-, we know that there exists $(\mathsf{tr}_S^-, \phi_S^-) \in \mathsf{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ with underlying substitution $\sigma_S^- = \sigma_S|_{dom(\sigma^-)}$, as well as two substitutions θ^- and λ^- such that:

- $dom(\theta^-) \subseteq \Sigma_{\mathsf{fresh}}$, for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in tr_S^- we have that $c \in dom(\theta^-)$, $c\theta^- \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S^-))$, and $\mathsf{w}' \prec^- c$ for any $\mathsf{w}' \in vars(c\theta^-)$ (where $\prec^-$ is the ordering induced by tr_S^-).
- for any $c \in dom(\theta^-)$, $(c\theta^-)\phi_S^-\!\downarrow$ is a Σ_0^+-message and it is an atom when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$.
- λ^- is the first-order substitution associated to θ^- through ϕ_S^-.
- $\phi^- = \phi_S^-\lambda^-$, $\sigma^- = \sigma_S^-\lambda^-$, and $(\mathsf{tr}_S^-\theta^-)\phi^-\!\downarrow = \mathsf{tr}^-\phi^-\!\downarrow$.

Let $c \in \Sigma_{\mathsf{fresh}}$ and $\mathsf{w} \in dom(\phi_S^-)$ such that c occurs in $\mathsf{w}\phi_S$. Then, as c does not occur in the Σ_0^--configuration $\mathcal{K}_P$, c must have been introduced in an input before the output of w. So $c \prec^- \mathsf{w}$. Let $\mathsf{w}' \in vars(c\theta^-)$. Then $\mathsf{w}' \prec^- c$ by definition of $\prec^-$. So $\mathsf{w}' \prec^- \mathsf{w}$ and the order induced by $\prec^-$ on $dom(\phi_S^-)$ satisfies the condition of Lemma 4.8 and Lemma 4.9. We distinguish three cases depending on α.

Case where $\alpha = \mathsf{phase}\ j$ for some integer j. In such a case, we have that $\mathcal{P}^+ = \mathcal{P}^-$, $\phi^+ = \phi^-$, $\sigma^+ = \sigma^-$, and $i^+ = j > i^-$. Let $\mathcal{P}_S^+ = \mathcal{P}_S^-$, $\phi_S^+ = \phi_S^-$, $\sigma_S^+ = \sigma_S^-$, and $i_S^+ = j$. Since $i_S^- = i^- < j$, we have that:

$$(\mathcal{P}_S^-; \phi_S^-; \sigma_S^-; i_S^-) \xRightarrow{\mathsf{phase}\ j} (\mathcal{P}_S^+; \phi_S^+; \sigma_S^+; j) = (\mathcal{P}_S^+; \phi_S^+; \sigma_S^+; i_S^+).$$

Considering $\theta^+ = \theta^-, \lambda^+ = \lambda^-$, and $\prec^+ = \prec^-$, it is easy to show that all our requirements are satisfied.

Case where $\alpha = \mathsf{out}(c, \mathsf{w}_0)$. In such a case, we have that $\mathcal{P}^- = \{\mathsf{out}(c, u).P_c\} \uplus \mathcal{Q}$ for some u, P_c, and $\mathcal{Q}$, $\mathcal{P}^+ = \{P_c\} \uplus \mathcal{Q}$, $\phi^+ = \phi^- \uplus \{\mathsf{w}_0 \triangleright u\sigma^-\}$, $\sigma^+ = \sigma^-$, and $i^+ = i^-$. Let $\mathcal{P}_S^+ = \{P_c\} \uplus \mathcal{Q}$, $\phi_S^+ = \phi_S^- \uplus \{\mathsf{w}_0 \triangleright u\sigma_S^-\}$, $\sigma_S^+ = \sigma_S^-$, and $i_S^+ = i_S^-$. We define $\theta^+ = \theta^-$ and $\lambda^+ = \lambda^-$.

First, we need to show that $u\sigma_S^-$ is a Σ_0^+-message. Thanks to our induction hypothesis, we know that $u\sigma^- = u\sigma_S^-\lambda^-$, and by hypothesis we have that $u\sigma^-$ is a Σ_0^--message. Thus, in order to conclude, we only have to show that $c\lambda^- \in \Sigma_0^-$ implies that $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$ for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in $u\sigma_S^-$. Let $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$ occurring in $u\sigma_S^-$. Since $u\sigma^- = u\sigma_S^-\lambda^-$, we know that $c \in dom(\lambda^-)$. Assume that $c\lambda^- \in \Sigma_0^-$ and let $x \in dom(\sigma) \setminus dom(mgu(\Gamma))$ be the unique variable such that $\rho(x) = c$. We have that $xmgu(\Gamma) = x$, thus we deduce that $c\lambda^- = ((xmgu(\Gamma))\rho)\lambda^- = x\sigma_S^-\lambda^- = x\sigma^-$. Hence, we have that $x\sigma \in \Sigma_0^-$, and thus $\rho(x) = c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$ by definition of the renaming ρ.

Therefore, we have shown that $u\sigma_S^-$ is a Σ_0^+-message, and thus we have that:

$$(\mathcal{P}_S^-; \phi_S^-; \sigma_S^-; i_S^-) \xRightarrow{\mathsf{out}(c, \mathsf{w}_0)} (\mathcal{P}_S^+; \phi_S^- \uplus \{\mathsf{w} \triangleright u\sigma_S^-\}; \sigma_S^-; i_S^-) = (\mathcal{P}_S^+; \phi_S^+; \sigma_S^+; i_S^+).$$

Considering $\theta^+ = \theta^-$, $\lambda^+ = \lambda^-$, and $\prec^+$ the extension of $\prec^-$ with $u \prec^+ \mathsf{w}_0$ for any $u \in \mathcal{W} \uplus \Sigma_{\mathsf{fresh}}$ occurring in tr_S^- or in $dom(\phi_0)$, it is easy to show that all our requirements are satisfied.

Case where $\alpha = \mathsf{in}(c, R)$. We have that $\mathcal{P}^- = \{\mathsf{in}(c, u).P_c\} \cup \mathcal{Q}$ for some $u \in \mathcal{T}_0(\Sigma_c, \Sigma_0^- \uplus \mathcal{N})$ and $R\phi^-\downarrow = (u\sigma^-)\tau$ for some τ with $dom(\tau) = vars(u\sigma^-)$. Moreover, we have that $\sigma^+ = \sigma^- \uplus \tau$, $(u\sigma^-)\tau = u\sigma^+$, $\phi^+ = \phi^-$, and $\mathcal{P}^+ = \{P_c\} \cup \mathcal{Q}$. We consider R_S minimal w.r.t. $\mu_{\phi_S^-}$ such that $(R_S\theta^-)\phi^-\downarrow = (u\sigma^-)\tau = u\sigma^+$. Note that such a R_S exists since $R_S = R$ is actually a candidate (but not necessary a minimal one). We can assume w.l.o.g. that R_S only use constants from Σ_{fresh} that have been introduced by tr_S^-, and thus that are in $dom(\theta^-)$.

Step 1: We prove that $R_S\phi_S^-\downarrow$ is a Σ_0^+-message. Assume that $R_S\phi_S^-\downarrow$ is not a Σ_0^+-message. We take the smallest subterm R' of R_S such that $R'\phi_S^-\downarrow$ is not a Σ_0^+-message. Let p be such that $R_S|_p = R'$. As $R'\phi_S^-\downarrow$ is not a Σ_0^+-message, we know that $R' \notin \Sigma_0^+ \uplus dom(\phi_S^-)$. Thus, we have that $R' = \mathsf{f}(R_1, \ldots, R_k)$ for some $\mathsf{f} \in \Sigma$. Moreover, by minimality of R', we know that $R_i\phi_S^-\downarrow$ is a Σ_0^+-message for $1 \leq i \leq k$. We now establish the following claim:

Claim. If $R_i\phi_S^-\downarrow \in \Sigma_{\mathsf{fresh}}$ for some $i \in \{1, \ldots, k\}$, then R_S is not minimal.

Proof. Assume $R_i\phi_S^-\downarrow = c \in \Sigma_{\mathsf{fresh}}$ for some $i \in \{1, \ldots, k\}$. Note that this implies that c occurs in ϕ_S^-, and thus c occurs in tr_S, and therefore $c \in dom(\theta^-)$. We consider $R_i' = c\theta^-$. Thanks to Lemma 4.9, we have that $(R_i\theta^-)\phi^-\downarrow = R_i\phi_S^-\downarrow\lambda^-$. We have that $R_i\phi_S^-\downarrow\lambda^- = c\lambda^- = (c\theta^-)(\phi_S^-\lambda^-)\downarrow$ since λ^- is the first-order substitution associated to θ^- through ϕ_S^-. Thus we have that $(R_i\theta^-)\phi^-\downarrow = (c\theta^-)\phi^-\downarrow = (R_i'\theta^-)\phi'\downarrow$. Let $\overline{R_S} = R_S[\mathsf{f}(R_1, \ldots, R_i', \ldots, R_k)]_p$. We have that $(\overline{R_S}\theta^-)\phi^-\downarrow = (R_S\theta^-)\phi^-\downarrow$. We have also that $\mu^1_{\phi_S^-}(R_i') < \mu^1_{\phi_S^-}(R_i)$ since $R_i'\phi_S^-\downarrow = (c\theta^-)\phi_S^-\downarrow$ is a Σ_0^+-message and it is an atom when $c \in \Sigma^{\mathsf{atom}}_{\mathsf{fresh}}$, and $R_i\phi_S^-\downarrow = c$. Thus, thanks to Lemma 4.6, we deduce that $\mu^1_{\phi_S^-}(\overline{R_S}) < \mu^1_{\phi_S^-}(R_S)$. Thus R_S is not minimal, and this proves the claim.

We now distinguish two cases depending on whether $\mathsf{f} \in \Sigma_{\mathsf{c}}$ or $\mathsf{f} \in \Sigma_{\mathsf{d}}$.

Case where $\mathsf{f} \in \Sigma_{\mathsf{c}}$. $R'\phi_S^-\downarrow$ is either not well-shaped or not well-sorted as it is not a Σ_0^+-message. If it is not well-sorted, then for one of its subrecipes R_i, $R_i\phi_S^-\downarrow$ is not an atom (it is well-sorted by minimality of R') while it should be. In particular, $(R_i\theta^-)\phi^-\downarrow = R_i\phi_S^-\downarrow\lambda^-$ is an atom. So $R_i\phi_S^-\downarrow$ must be in Σ_{fresh}, which will contradict the minimality of R_S thanks to our claim. We deduce that $R'\phi_S^-\downarrow$ is well-sorted.

Now, we assume that $R'\phi_S^-\downarrow$ is not well-shaped, we consider the shape of f, $sh_{\mathsf{f}} = \mathsf{f}(s_1, \ldots, s_k)$ for some $s_1, \ldots, s_k$. As $R'\phi_S^-\downarrow$ has a bad shape and is a f-term, there must be a j such that $R_j\phi_S^-\downarrow$ is not an instance of s_j. In particular, s_j is not a variable and we have that $s_j = sh_{\mathsf{g}}$ for some function symbol g (thanks to the compatibility of the shapes). We know that $R_j\phi_S^-\downarrow$ is a Σ_0^+-message and thus we have that $(R_j\theta^-)\phi^-\downarrow = (R_j\phi_S^-\downarrow)\lambda^-$ (thanks to Lemma 4.9) is an instance of s_j as $(R\theta^-)\phi^-\downarrow$ is a Σ_0^--message. Relying on our claim, we know that $R_j\phi_S^-\downarrow \notin \Sigma_{\mathsf{fresh}}$, thus $R_j\phi_S^-\downarrow$ is a g-term, and since we know that $R_j\phi_S^-\downarrow$ is a Σ_0^+-message, we know that it is an instance of s_j, yielding to a contradiction.

Case where $\mathsf{f} = \mathsf{des} \in \Sigma_{\mathsf{d}}$. In such a case, we have that $R' = \mathsf{des}(R_1, \ldots, R_k)$. Let $\ell_{\mathsf{des}} = \mathsf{des}(t_1, t_2, \ldots, t_k)$. We distinguish two subcases.

First, we assume that there is some $i \in \{1, \ldots, k\}$ such that $R_i\phi_S^-\downarrow$ does not unify with t_i. As $R_i\phi_S^-\downarrow$ is a Σ_0^+-message, it has a good shape. Thus, the only way to not unify with the linear term t_i whereas $R_i\phi_S^-\downarrow\lambda^- = (R_i\theta^-)\phi^-\downarrow$ (thanks to Lemma 4.9) does, is when $R_i\phi_S^-\downarrow = c$ for some $c \in \Sigma_{\mathsf{fresh}}$. Relying on our claim, we obtain a contradiction.

Second, we assume that $R_i\phi_S^-\downarrow$ unifies with t_i for each $i \in \{1, \ldots k\}$. In such a case, we know that $\ell_{\mathsf{des}} = \mathsf{des}(t_1, t_2, \ldots, t_k)$ is a non-linear term and we denote x the non linear variable occurring in ℓ_{des}. Let $I_0 = \{1 \leq i \leq k \mid x \text{ occurs in } t_i\}$. We know that $1 \in I_0$. For any $i \in I_0$, we denote p_i the position in t_i such that $t_i|_{p_i} = x$.

Since $R'\phi_S^-\downarrow\lambda^-$ is a Σ_0^--message, we know that $t = \mathsf{des}(R_1\phi_S^-\downarrow\lambda^-, \ldots, R_k\phi_S^-\downarrow\lambda^-)$ unifies with $\mathsf{des}(t_1, \ldots, t_k)$. Therefore, we know that there exists an atomic Σ_0^--message a such that $t|_{i.p_i} = a$ for any $i \in I_0$. We know that $R'\phi_S^-\downarrow$ is not a Σ_0^+-message whereas $R_i\phi_S^-\downarrow$ are Σ_0^+-message. Thus, we have that $t_S = \mathsf{des}(R_1\phi_S^-\downarrow, \ldots, R_k\phi_S^-\downarrow)$ does not unify with $\mathsf{des}(t_1, \ldots, t_k)$. We deduce that for any $i \in I_0$, we have that either $t_S|_{i.p_i} = a$, or $t_S|_{i.p_i} = c$ for some $c \in \Sigma_{\mathsf{fresh}}$ such that $c\lambda^- = a$. We distinguish two cases depending on whether $R_1\phi_S^-\downarrow|_{p_1} = c'$ for some $c' \in \Sigma_{\mathsf{fresh}}$ or not.

First, we assume that $R_1\phi_S^-\downarrow|_{p_1} = c'$ for some $c' \in \Sigma_{\mathsf{fresh}}$. Let ν_{des} be the substitution such that $x\nu_{\mathsf{des}} = c'$ and $y\nu_{\mathsf{des}} = c_{min}$ for any other variable $y \in vars(\ell_{\mathsf{des}})$. Let $\overline{R'} = \mathsf{des}(R_1, t_2\nu_{\mathsf{des}}, \ldots, t_k\nu_{\mathsf{des}})$. Actually, we have that $\overline{R'}\phi_S^-\downarrow$ is a Σ_0^+-message, and thanks to Lemma 4.9, we know that $(\overline{R'}\theta^-)\phi^-\downarrow = \overline{R'}\phi_S^-\downarrow\lambda^-$. Since $\overline{R'}\phi_S^-\downarrow$ is a Σ_0^+-message, we have that $\mu^1_{\phi_S^-}(\overline{R'}) < \{\mathsf{des}\}$, and thus $\mu^1_{\phi_S^-}(\overline{R'}) < \mu^1_{\phi_S^-}(R')$. We also deduce that $(\overline{R'}\theta^-)\phi^-\downarrow$ is a Σ_0^--message, and we have that $(\overline{R'}\theta^-)\phi^-\downarrow = (R'\theta^-)\phi^-\downarrow$ since the Σ_0^+-recipes $\overline{R'}$ and R' coincide on their first argument.

Second, we assume that $R_1\phi_S^-\downarrow|_{p_1} \notin \Sigma_{\mathsf{fresh}}$, i.e. $R_1\phi_S^-\downarrow|_{p_1} = a$. We know that there exists $i_0 \in I_0$ such that $R_{i_0}\phi_S^-\downarrow|_{p_{i_0}} = c'$ for some $c' \in \Sigma_{\mathsf{fresh}}$ (otherwise t_S will reduce), and we have that $c'\lambda^- = a$. Let ν_{des} be the substitution such that $x\nu_{\mathsf{des}} = c'\theta^-$ and $y\nu_{\mathsf{des}} = c_{min}$ for any other variable $y \in vars(\ell_{\mathsf{des}})$. Let $\overline{R'} = \mathsf{des}(R_1, t_2\nu_{\mathsf{des}}, \ldots, t_k\nu_{\mathsf{des}})$. As $\mu^1_{\phi_S^-}(c'\theta^-) < \mu^1_{\phi_S^-}(c')$, we have that $\mu^1_{\phi_S^-}(R_1) \uplus \mu^1_{\phi_S^-}(t_2\nu_{\mathsf{des}}) \uplus \ldots \uplus \mu^1_{\phi_S^-}(t_k\nu_{\mathsf{des}}) < \mu^1_{\phi_S^-}(R_1) \uplus \ldots \uplus \mu^1_{\phi_S^-}(R_k)$. Therefore, relying on Lemma A.3, we deduce that:

$$\begin{array}{rcl}
\mu^1_{\phi_S^-}(\overline{R'}) & = & \mathsf{Multi}(\overline{R'}\phi_S^-\downarrow) \\
& \leq & \mathsf{Multi}(\mathsf{des}(R_1\phi_S^-\downarrow, t_2\nu_{\mathsf{des}}\phi_S^-\downarrow, \ldots, t_k\nu_{\mathsf{des}}\phi_S^-\downarrow)) \\
& = & \{\mathsf{des}\} \uplus \mu^1_{\phi_S^-}(R_1) \uplus \mu^1_{\phi_S^-}(t_2\nu_{\mathsf{des}}) \uplus \cdots \uplus \mu^1_{\phi_S^-}(t_k\nu_{\mathsf{des}}) \\
& < & \{\mathsf{des}\} \uplus \mu^1_{\phi_S^-}(R_1) \uplus \cdots \uplus \mu^1_{\phi_S^-}(R_k) \\
& = & \mu^1_{\phi_S^-}(R')
\end{array}$$

Since $R_1\phi_S^-\downarrow$ is a Σ_0^+-message, we have that $\mathsf{des}((R_1\theta^-)\phi^-\downarrow, ((t_2\nu_{\mathsf{des}})\theta^-)\phi^-\downarrow, \ldots, ((t_k\nu_{\mathsf{des}})\theta^-)\phi^-\downarrow) = \mathsf{des}((R_1\phi_S^-)\downarrow\lambda^-, ((t_2\nu_{\mathsf{des}})\theta^-)\phi^-\downarrow, \ldots, ((t_k\nu_{\mathsf{des}})\theta^-)\phi^-\downarrow)$. Such a term unifies with ℓ_{des} since we have that $R_1\phi_S^-\downarrow\lambda^-|_{p_1} = R_1\phi_S^-\downarrow|_{p_1}\lambda^- = a\lambda^- = a$, and for any $i \in I_0$, we have that:

$$((t_i\nu_{\mathsf{des}})\theta^-)\phi^-\downarrow|_{p_i} = (t_i\nu_{\mathsf{des}})\phi^-\downarrow|_{p_i} = ((t_i\nu_{\mathsf{des}})|_{p_i})\phi^-\downarrow = (c'\theta^-)\phi^-\downarrow = c'\lambda^- = a.$$

Thus, we have that $(\overline{R'}\theta^-)\phi^-\downarrow$ is a Σ_0^--message, and we have that $(\overline{R}'\theta^-)\phi^-\downarrow = (R'\theta^-)\phi^-\downarrow$ since the Σ_0^+-recipes $\overline{R'}$ and R' coincide on their first argument.

In both cases, we have seen that $(\overline{R'}\theta^-)\phi^-\downarrow = (R'\theta^-)\phi^-\downarrow$, and also that $\mu^1_{\phi_S^-}(\overline{R'}) < \mu^1_{\phi_S^-}(R')$. Since we know that $R'\phi_S^-\downarrow$ is not a Σ_0^+-message, Lemma 4.6 applies, we obtain that $\mu^1_{\phi_S^-}(R_S[\overline{R'}]_p) < \mu^1_{\phi_S^-}(R_S[R']_p) = \mu^1_{\phi_S^-}(R_S)$, and this contradicts the minimality of R_S.

Step 2: We now prove that R_S is a simple Σ_0^+-recipe, in normal form w.r.t. $\downarrow$, and of the form $C[R_1, \ldots, R_n]$ where for each $i \in \{1, \ldots, n\}$, we have that $R_i\phi_S^-\downarrow$ is either an encrypted term, or a name from $\mathcal{N}$, or a constant from Σ_0^+.

Assume R_S is *not* in normal form w.r.t. $\downarrow$. By Lemma 4.7, we know that $\mu^1_{\phi_S^-}(R_S\downarrow) < \mu^1_{\phi_S^-}(R_S)$. Moreover, as $R_S\theta^- \twoheadrightarrow^* R_S\downarrow\theta^-$, Lemma 4.3 applies: $(R_S\downarrow\theta^-)\phi^-\downarrow = (R_S\theta^-)\phi^-\downarrow$, and this contradicts the minimality of R_S. Therefore, we know that R_S is in normal form w.r.t. $\downarrow$, and thus R_S is a simple Σ_0^+-recipe thanks to Lemma 4.5. Therefore, we have that $R_S = C[R_1, \ldots, R_n]$ where each R_i is a subterm recipe with $1 \leq i \leq n$. If there is $i_0 \in \{1, \ldots, n\}$ such that $\mathsf{root}(R_{i_0}\phi_S^-\downarrow) = \mathsf{f} \in \Sigma_c$ and f is a transparent function symbol, then $R_{i_0}\phi_S^-\downarrow = \mathsf{f}(C_1^{\mathsf{f}}[R_{i_0}], \ldots, C_k^{\mathsf{f}}[R_{i_0}])\phi_S^-\downarrow$. We consider the context $\overline{C}$ such

that $\overline{C}[R_1,\ldots,R_n] = C[R_1,\ldots,\mathrm{f}(C_1^{\mathrm{f}}[R_{i_0}],\ldots,C_k^{\mathrm{f}}[R_{i_0}]),\ldots,R_n]$. We have that $R'_S = \overline{C}[R_1,\ldots,R_n]$ is a Σ_0^+-recipe such that $R'_S\phi_S^-\downarrow = R_S\phi_S^-\downarrow$, and thus $\mu^1_{\phi_S}(R_S) = \mu^1_{\phi_S}(R'_S)$, and $(R'_S\theta^-)\phi^-\downarrow = R'_S\phi_S^-\downarrow\lambda^- = R_S\phi_S^-\downarrow\lambda^- = (R_S\theta^-)\phi^-\downarrow$. We have that $\mu^2_{\phi_S}(R'_S) < \mu^2_{\phi_S}(R_S)$ so this contradicts the minimality of R_S. Therefore, we deduce that each $R_i\phi_S\downarrow$ (with $1 \le i \le n$) is either an encrypted term, a constant from Σ_0^+, or a name from $\mathcal{N}$.

Step 3. Let $\mathcal{P}_S^+ = \mathcal{P}^+$, $\phi_S^+ = \phi_S^-$, $\sigma_S^+ = \sigma_S|_{dom(\sigma^+)}$, and $i_S^+ = i_S^-$. We are going to show that there exists a Σ_0^+-recipe $\overline{R_S}$ such that:

$$(\mathcal{P}_S^-;\phi_S^-;\sigma_S^-;i_S^-) \xrightarrow{\mathsf{in}(c,\overline{R_S})} (\mathcal{P}_S^+;\phi_S^+;\sigma_S^+;i_S^+)$$

as well as two substitutions θ^+ and λ^+ that satisfy all our requirements.

If $R_S\phi_S^-\downarrow = u\sigma_S^+$, then let $\overline{R_S} = R_S$, $\theta^+ = \theta^-$, and $\lambda^+ = \lambda^-$. To conclude, it remains to establish that all our requirements are satisfied. In particular we have to show that *(i)* $(\overline{R_S}\theta^+)\phi^+\downarrow = R\phi^+\downarrow$, and *(ii)* $\sigma^+ = \sigma_S^+\lambda^+$.

(i) We have that $(\overline{R_S}\theta^+)\phi^+\downarrow = (R_S\theta^-)\phi^-\downarrow = u\sigma^+ = R\phi^-\downarrow = R\phi^+\downarrow$.

(ii) Let $Z = vars(u\sigma^-) = dom(\tau)$. We have that $\overline{R_S}\phi_S^-\downarrow = u\sigma_S^+ = u(\sigma_S^- \uplus \sigma_S|_Z)$ is a Σ_0^+-message, and $(R_S\theta^-)\phi^-\downarrow = R\phi^-\downarrow = u\sigma^+ = u(\sigma^- \uplus \sigma|_Z)$. By Lemma 4.9, we know that $\overline{R_S}\phi_S^-\downarrow\lambda^- = (R_S\theta^-)\phi^-\downarrow$, and thus $u(\sigma_S^-\lambda^- \uplus \sigma_S|_Z\lambda^-) = u(\sigma^- \uplus \sigma|_Z)$. This allows us to conclude that $\sigma_S|_Z\lambda^- = \sigma|_Z$, and thus we have that:

$$\sigma_S^+\lambda^+ = (\sigma_S^- \uplus \sigma_S|_Z)\lambda^- = \sigma_S^-\lambda^- \uplus \sigma_S|_Z\lambda^- = \sigma^- \uplus \sigma|_Z = \sigma^- \uplus \tau = \sigma^+.$$

Therefore, from now on, we assume that $R_S\phi_S^-\downarrow \neq u\sigma_S^+$. Let A be the set of fresh constants that occur until this execution step, i.e. $A = St(img(\sigma_S^+)) \cap \Sigma_{\mathsf{fresh}}$. We define $\lambda^+ = \lambda|_A$. As $\sigma = \sigma_S\lambda$, we deduce $\sigma^+ = \sigma_S^+\lambda^+$. Since $R_S\phi_S^-\downarrow$ is a Σ_0^+-message, relying on Lemma 4.9, we deduce that $R_S\phi_S^-\downarrow\lambda^+ = R_S\phi_S^-\downarrow\lambda^- = (R_S\theta^-)\phi^-\downarrow = R\phi^-\downarrow = u\sigma^+ = (u\sigma_S^+)\lambda^+$. Let $t = R_S\phi_S\downarrow$ and $v = u\sigma_S^+$. We have that $t \neq v$ and $t\lambda^+ = v\lambda^+$.

Since $t \neq v$, we know that there exists a position p defined in t and v such that $\mathrm{root}(t|_p) \neq \mathrm{root}(v|_p)$. Let p be any position defined in t and v such that $\mathrm{root}(t|_p) \neq \mathrm{root}(v|_p)$. Since $t\lambda^+ = v\lambda^+$ and $dom(\lambda^+) \subseteq \Sigma_{\mathsf{fresh}}$, we have that $t|_p \in \Sigma_{\mathsf{fresh}}$ or $v|_p \in \Sigma_{\mathsf{fresh}}$.

We first assume that there exists such a position p that falls outside the context C. More precisely, we have that $p = p'.p''$ (with p' a strict prefix of p) and $C[R_1,\ldots,R_n]|_{p'} = R_{i_0}$ for some $i_0 \in \{1,\ldots,n\}$. Therefore, since R_{i_0} is a subterm-recipe, we know that $t|_{p'} = R_{i_0}\phi_S^-\downarrow$ is an encrypted subterm of ϕ_S^- Relying on Lemma 4.10 and denoting $\mathcal{K}_S^- = (\mathcal{P}_S^-;\phi_S^-;\sigma_S^-;i_S^-)$, we have that:

- $t|_{p'} \in ESt(\phi_S^-) \subseteq ESt(\mathcal{K}_S^-\sigma_S^-) \subseteq ESt(\mathcal{K}_P\sigma_S^-) = ESt(\mathcal{K}_P(mgu(\Gamma)|_{dom(\sigma_S^-)}\rho)) \subseteq ESt(\mathcal{K}_P)\sigma_S^-$.
- $v|_{p'} = u\sigma_S^+|_{p'} \in ESt(\mathcal{K}_P\sigma_S^+) \subseteq ESt(\mathcal{K}_P(mgu(\Gamma)|_{dom(\sigma_S^+)})\rho) \subseteq ESt(\mathcal{K}_P)\sigma_S^+$.

This allows us to conclude that there exist $t', v' \in ESt(\mathcal{K}_0)$ such that $t'\sigma_S^- = t|_{p'}$, and $v'\sigma_S^+ = v|_{p'}$. Since $t|_{p'}$ is a Σ_0^+-message, we also know that $t'\sigma_S^+ = t|_{p'}$. Since $t\lambda^+ = v\lambda^+$, we know that $t|_{p'}\lambda^+ = v|_{p'}\lambda^+$, and thus $(t'\sigma_S^+)\lambda^+ = (v'\sigma_S^+)\lambda^+$. We have that $(t'\sigma_S^+)\lambda^+ = t'(\sigma_S^+\lambda^+)$ and also that $(v'\sigma_S^+)\lambda^+ = v'(\sigma_S^+\lambda^+)$. Since we have that $\sigma^+ = \sigma_S^+\lambda^+$, we deduce that $t'\sigma^+ = v'\sigma^+$, and thus $t'\sigma = v'\sigma$. By definition of σ_S, we have that $t'\sigma_S = v'\sigma_S$. We know that $t'\sigma_S^- = t|_{p'}$, and since $t|_{p'}$ is ground, we deduce that $t'\sigma_S = t|_{p'}$. Similarly, we have that $v'\sigma_S^+ = v|_{p'}$, and since $v|_{p'}$ is ground, we deduce that $v'\sigma_S = v|_{p'}$. Thus, we have that $t|_{p'} = v|_{p'}$ leading to a contradiction since we have assumed that t and v differ below the position p'.

Now, we know that for any position p defined in t and v such that $\mathrm{root}(t|_p) \neq \mathrm{root}(v|_p)$, we have that $t|_p$ or $v|_p$ is in Σ_{fresh}, and p is a position of C. If $t|_p = c \in \Sigma_{\mathsf{fresh}}$, let $R'_S = R_S[c\theta^-]_p$, we have

that $\mu^1_{\phi^-_S}(R'_S) < \mu^1_{\phi^-_S}(R_S)$ since $\mu^1_{\phi^-_S}(c\theta^-) < \mu^1_{\phi^-_S}(R_S|_p)$ (note that $R_S|_p\phi^-_S\downarrow = c$ whereas $(c\theta^-)\phi^-_S\downarrow$ is a Σ^+_0-message and it is an atom when $c \in \Sigma^{\mathsf{atom}}_{\mathsf{fresh}}$). Moreover, we have that $(R'_S\theta^-)\phi^-\downarrow = (R_S\theta^-)\phi^-\downarrow$. This will contradict the minimality of R_S. Therefore, we have that $t|_p \notin \Sigma_{\mathsf{fresh}}$, and thus $v|_p = c$ for some $c \in \Sigma_{\mathsf{fresh}}$.

Let $p_1, \ldots, p_m$ be the positions such that for each $i \in \{1, \ldots, m\}$, we have that p_i is defined in both t and v, and $\mathrm{root}(t|_{p_i}) \neq \mathrm{root}(v|_{p_i})$. For each $i \in \{1, \ldots, m\}$, we know that p_i is a position of C such that $t|_{p_i} \notin \Sigma_{\mathsf{fresh}}$, and $v|_{p_i} \in \Sigma_{\mathsf{fresh}}$. We denote c^i_{fresh} the constant from Σ_{fresh} such that $v|_{p_i} = c^i_{\mathsf{fresh}}$. Note that it may happen that $c^i_{\mathsf{fresh}} = c^j_{\mathsf{fresh}}$ for some $i \neq j$. Let $\overline{C}$ be the context obtained from C by putting c^1_{fresh} at position p_1, c^2_{fresh} at position p_2, ... Let $\overline{R_S} = \overline{C}[R_1, \ldots, R_n]$. We have that $\overline{R_S}\phi^-_S\downarrow = u\sigma^+_S$ by construction. Let $R_{p_i} = (C[R_1, \ldots, R_n]|_{p_i})\theta^-$ for each $i \in \{1, \ldots, m\}$. Thanks to Lemma 4.9, we have that $R_{p_i}\phi^-\downarrow = R_{p_i}\phi^-_S\downarrow\lambda^-$ for each $i \in \{1, \ldots, m\}$. As for each $i \in \{1, \ldots, m\}$, we have that $R_{p_i}\phi^-_S\downarrow\lambda^- = u\sigma^+_S\lambda^+|_{p_i}$, we get that $R_{p_i}\phi^-_S\downarrow\lambda^- = c^i_{\mathsf{fresh}}\lambda^+$ for each $i \in \{1, \ldots, m\}$. Note that in case $c^i_{\mathsf{fresh}} = c^j_{\mathsf{fresh}}$, we have that $R_{p_i}\phi^-\downarrow = R_{p_j}\phi^-\downarrow$. Let $\theta^+ = \theta^- \uplus \{c^i_{\mathsf{fresh}} \mapsto R_{p_i} \mid c^i_{\mathsf{fresh}} \notin dom(\theta^-)\}$. In case we have that $c^i_{\mathsf{fresh}} = c^j_{\mathsf{fresh}}$ for some $i \neq j$, we choose arbitrarily between R_{p_i} and R_{p_j}.

Now, it remains to establish that all our requirements are satisfied. In particular, we have to show that:

(i) $\sigma^+_S = \sigma^-_S \uplus \tau_S$ for some τ_S such that $dom(\tau_S) = vars(u\sigma^-_S)$ and $\overline{R_S}\phi^-_S\downarrow = (u\sigma^-_S)\tau_S$.
(ii) for any $c \in dom(\theta^+) \setminus dom(\theta^-)$, $(c\theta^+)\phi^-_S\downarrow$ is a Σ^+_0-message, and it is an atom when $c \in \Sigma^{\mathsf{atom}}_{\mathsf{fresh}}$.
(iii) for any $c \in dom(\theta^+) \setminus dom(\theta^-)$, we have that $c\lambda^+ = (c\theta^+)(\phi^-_S\lambda^-)\downarrow$.
(iv) $(\overline{R_S}\theta^+)\phi^+\downarrow = R\phi^+\downarrow$.

We prove each item one by one.

(i) Let $\tau_S = \sigma_S|_{dom(\tau)}$. We have that $dom(\tau_S) = dom(\tau) = vars(u\sigma^-) = vars(u\sigma^-_S)$. We have shown that $\overline{R_S}\phi^-_S\downarrow = u\sigma^+_S$, and thus $\overline{R_S}\phi^-_S\downarrow = u(\sigma^-_S \uplus \tau_S) = (u\sigma^-_S)\tau_S$.
(ii) Let $c \in dom(\theta^+) \setminus dom(\theta^-)$. We have that $c = c^i_{\mathsf{fresh}}$ for some $i \in \{1, \ldots, m\}$, and $(c^i_{\mathsf{fresh}}\theta^+)\phi^-_S\downarrow = R_{p_i}\phi^-_S\downarrow$ is a Σ^+_0-message. Assume that $c^i_{\mathsf{fresh}} \in \Sigma^{\mathsf{atom}}_{\mathsf{fresh}}$. In such a case, there exists $x_i \in dom(\rho)$ such that $x_i\rho = c^i_{\mathsf{fresh}}$. As $x_i\rho$ is atomic, we know that $x_i\sigma$ is atomic (by definition of ρ), and we have that $x_i\sigma = x_i\sigma_S\lambda$. Since $\sigma_S = mgu(\Gamma)\rho$, we have that $x_i\sigma = x_i mgu(\Gamma)\rho\lambda$. As $x_i \in dom(\rho)$, $x_i \notin dom(mgu(\Gamma))$ by definition of ρ. So $x_i\sigma = x_i\rho\lambda = c^i_{\mathsf{fresh}}\lambda = c^i_{\mathsf{fresh}}\lambda^+$. Since we have shown that $x_i\sigma$ is atomic, we deduce that $c^i_{\mathsf{fresh}}\lambda^+$ is atomic. Thus, since we have shown that $R_{p_i}\phi^-_S\downarrow\lambda^- = c^i_{\mathsf{fresh}}\lambda^+$, we know that $R_{p_i}\phi^-_S\downarrow\lambda^-$ is atomic which implies that $R_{p_i}\phi^-_S\downarrow$ is either atomic or a $c \in \Sigma_{\mathsf{fresh}}$ with $c \in dom(\lambda^-)$. So we can assume $c \in \Sigma_{\mathsf{fresh}}$ with $c \in dom(\lambda^-)$. There is a $x \in dom(\rho)$ such that $x\rho = c$. We have $x\sigma = x\sigma_S\lambda = x mgu(\Gamma)\rho\lambda$. As $x \in dom(\rho)$, $x \notin dom(mgu(\Gamma))$ and $x\sigma = x\rho\lambda = c\lambda = c\lambda^-$. As $x\sigma$ is atomic, $x\rho$ is atomic (by definition of ρ), so c is atomic, and thus $R_{p_i}\phi^-_S\downarrow$ is atomic.
(iii) Let $c \in dom(\theta^+) \setminus dom(\theta^-)$. We have that $c = c^i_{\mathsf{fresh}}$ for some $i \in \{1, \ldots, m\}$, and $c^i_{\mathsf{fresh}}\lambda^+ = R_{p_i}\phi^-\downarrow = R_{p_i}(\phi^-_S\lambda^-)\downarrow = (c^i_{\mathsf{fresh}}\theta^+)(\phi^-_S\lambda^-)\downarrow$.
(iv) $(\overline{R_S}\theta^+)\phi^+\downarrow = (R_S\theta^-)\phi^-\downarrow = R\phi^-\downarrow = R\phi^+\downarrow$.

This concludes the case where α is an input, and we get the result. □

We can conclude with the proof of our main theorem on reachability properties.

Theorem 3.8. *Let $\mathcal{K}_P$ be a Σ^-_0-configuration type-compliant w.r.t. $(\mathcal{T}_0, \delta_0)$. If $\mathcal{K}_P \overset{\mathsf{tr}}{\Longrightarrow} (\mathcal{P}; \phi; \sigma; i)$ w.r.t. Σ^-_0 then there exists a well-typed execution $\mathcal{K}_P \overset{\mathsf{tr}'}{\Longrightarrow} (\mathcal{P}; \phi'; \sigma'; i)$ w.r.t. Σ^+_0 such that $\overline{\mathsf{tr}'} = \overline{\mathsf{tr}}$.*

Conversely, if $\mathcal{K}_P \overset{\mathsf{tr}'}{\Longrightarrow} (\mathcal{P};\phi';\sigma';i)$ *is a well-typed execution w.r.t.* Σ_0^+*, then there exists* $\mathcal{K}_P \overset{\mathsf{tr}}{\Longrightarrow} (\mathcal{P};\phi;\sigma;i)$ *w.r.t.* Σ_0^- *such that* $\overline{\mathsf{tr}} = \overline{\mathsf{tr}'}$.

Proof. As explained in Section 3.3.1, the converse part is obvious. Thus, we now prove the direct part. Let $\mathcal{K}_P$ be a Σ_0^--configuration type-compliant w.r.t. $(\mathcal{T}_0, \delta_0)$. Assume $\mathcal{K}_P \overset{\mathsf{tr}}{\Longrightarrow} (\mathcal{P};\phi;\sigma;i)$ w.r.t. Σ_0^-. Thanks to Proposition 4.11, we know that there exists $(\mathsf{tr}_S, \phi_S) \in \mathsf{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ with underlying substitution σ_S with $dom(\sigma_S) = dom(\sigma)$ such that $\sigma_S = mgu(\Gamma)\rho$, as well as two substitutions λ and θ such that:

- $\Gamma = \{(u, v) \mid u, v \in ESt(\mathcal{K}_P)$ such that $u\sigma = v\sigma\}$.
- ρ is a bijective renaming from variables in $dom(\sigma) \setminus dom(mgu(\Gamma))$ to constants in Σ_{fresh} such that $x\rho \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$ if, and only if, $x\sigma$ is an atomic Σ_0^--message.
- $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$, for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in tr_S, we have that $c \in dom(\theta)$, $c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S))$, and $\mathsf{w}' \prec c$ for any $\mathsf{w}' \in vars(c\theta)$ (where $\prec$ is the ordering induced by tr_S).
- for any $c \in dom(\theta)$, $(c\theta)\phi_S{\downarrow}$ is a Σ_0^+-message and it is an atom when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$.
- λ is the first-order substitution associated to θ through ϕ_S.
- $\phi = \phi_S\lambda$, $\sigma = \sigma_S\lambda$, and $(\mathsf{tr}_S\theta)\phi{\downarrow} = \mathsf{tr}\phi{\downarrow}$.

Since $(\mathsf{tr}_S\theta)\phi{\downarrow} = \mathsf{tr}\phi{\downarrow}$, we have that $\overline{\mathsf{tr}_S} = \overline{\mathsf{tr}}$. Since we have enough constants of each type, we may assume w.l.o.g. that ρ is well-typed Since $\mathcal{K}_P$ is type-compliant, we know that encrypted subterms in $ESt(\mathcal{K}_P)$ which are unifiable have the same type. Then, by definition of a typing system, this allows us to deduce that $mgu(\Gamma)$ is well-typed, and thus $\sigma_S = mgu(\Gamma)\rho$ is well-typed. This means that (tr_S, ϕ_S) is well-typed and concludes the proof. □

B PROOF OF SECTION 5

The purpose of the lemma below is to lift a "symbolic" trace (i.e. a trace using special constants from Σ_{fresh}) into a concrete one. We are allowed to replace any constant from Σ_{fresh} by any term provided that such a term is deducible at the time we need it. Moreover, we have to pay attention to preserve atomicity. Under these conditions, we can show that the resulting trace is still a valid one.

Lemma B.1. *Let* $\mathcal{K}_Q = (Q_0;\psi_0;\sigma_0;i_0)$ *be a* Σ_0^-*-configuration, and* $(\mathsf{tr}_S, \psi_S) \in \mathsf{trace}_{\Sigma_0^+}(\mathcal{K}_Q)$*. Let* θ *be a substitution such that* $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$*, for any* $c \in \Sigma_{\mathsf{fresh}}$ *occurring in* tr_S*, we have that* $c \in dom(\theta)$*,* $c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\psi_S))$*, and* $w' \prec c$ *for any* $w' \in vars(c\theta)$ *(where* $\prec$ *is the ordering induced by* tr_S*). We also assume that* $(c\theta)\psi_S{\downarrow}$ *is a* Σ_0^+*-message for any* $c \in dom(\theta)$ *and an atomic one in case* $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$*. We have that* $(\mathsf{tr}_S\theta, \psi_S\lambda) \in \mathsf{trace}_{\Sigma_0^-}(\mathcal{K}_Q)$ *where* λ *is the first-order substitution associated to* θ *through* ψ_S*.*

Proof. Since $(\mathsf{tr}_S, \psi_S) \in \mathsf{trace}_{\Sigma_0^+}(\mathcal{K}_Q)$, we know that $\mathcal{K}_Q = (Q_0;\psi_0;\sigma_0;i_0) \overset{\mathsf{tr}_S}{\Longrightarrow} (Q_S;\psi_S;\sigma_S;i_S)$. Given θ and λ as defined in the lemma, we establish that: $\mathcal{K}_Q = (Q_0;\psi_0;\sigma_0;i_0) \overset{\mathsf{tr}_S\theta}{\Longrightarrow} (Q;\psi;\sigma;i)$ where $Q = Q_S$, $\phi = \psi_S\lambda$, $\sigma = \sigma_S\lambda$, and $i = i_S$. We show this result by induction on the length of the trace.

Base case: tr is empty. Let $(Q;\psi;\sigma;i) = \mathcal{K}_Q$. Since $\mathcal{K}_Q$ is a Σ_0^--configuration, we have that $\psi\lambda = \psi$, and $\sigma\lambda = \sigma$. Therefore, the result trivially holds.

Inductive case: $\mathsf{tr} = \mathsf{tr}^-.\alpha_S$. In such a case, we have that:

$$\mathcal{K}_Q = (Q_0;\psi_0;\sigma_0;i_0) \overset{\mathsf{tr}_S}{\Longrightarrow} (Q_S^-;\psi_S^-;\sigma_S^-;i_S^-) \overset{\alpha_S}{\Longrightarrow} (Q_S;\psi_S;\sigma_S;i_S)$$

Thanks to our induction hypothesis applied on tr^-, we know that:

$$\mathcal{K}_Q = (Q_0;\psi_0;\sigma_0;i_0) \overset{\mathsf{tr}_S\theta}{\Longrightarrow} (Q^-;\psi^-;\sigma^-;i^-)$$

with $\mathcal{Q}^- = \mathcal{Q}_S^-$, $\psi^- = \psi_S^-\lambda$, $\sigma^- = \sigma_S^-\lambda$, and $i^- = i_S^-$. We distinguish three cases depending on α_S.

- *Case* $\alpha_S = \mathsf{phase}\, j$ *for some integer* j. In such a case, we have that $\mathcal{Q}_S = \mathcal{Q}_S^-$, $\psi_S = \psi_S^-$, $\sigma_S = \sigma_S^-$, and $i_S = j > i_S^-$. Let $\mathcal{Q} = \mathcal{Q}^-$, $\psi = \psi^-$, $\sigma = \sigma^-$, $i = j$. Since $i^- = i_S^- < j$. We have that:

$$(\mathcal{Q}^-; \psi^-; \sigma^-; i^-) \xrightarrow{\mathsf{phase}\, j} (\mathcal{Q}^-; \psi^-; \sigma^-; j) = (\mathcal{Q}; \psi; \sigma; i)$$

  We have that $\mathcal{Q} = \mathcal{Q}_S$, $\psi = \psi_S\lambda$, $\sigma = \sigma_S\lambda$, and $i = i_S$. It gives us the result.
- *Case* $\alpha_S = \mathsf{in}(c, R)$ *for some* Σ_0^+*-recipe* R. In such a case, we have that $\mathcal{Q}_S^- = \{\mathsf{in}(c, u).Q_c\} \uplus \mathcal{P}$, and we have also that $u\sigma_S^-$ and $R\psi_S^-\downarrow$ (ground term) are unifiable with some substitution τ, and we have that $\sigma_S = \sigma_S^- \uplus \tau$. Thanks to Lemma 4.8, we can apply Lemma 4.9, and thus we have:

$$(R\theta)\psi^-\downarrow = (R\theta)(\psi_S\lambda)\downarrow = (R\psi_S^-\downarrow)\lambda = [(u\sigma_S^-)\tau]\lambda = u(\sigma_S^- \uplus \tau)\lambda = (u(\sigma_S^-\lambda))(\tau\lambda).$$

  Let $\mathcal{Q} = \{Q_c\} \uplus \mathcal{P}$, $\psi = \psi^-$, $\sigma = \sigma^- \uplus \tau\lambda$, and $i = i^-$. We have that:

$$(\{\mathsf{in}(c, u).Q_c\} \uplus \mathcal{P}; \psi^-; \sigma^-; i^-) \xrightarrow{\mathsf{in}(c, R\theta)} (\{Q_c\} \uplus \mathcal{P}; \psi^-; \sigma^- \uplus \tau\lambda; i^-) = (\mathcal{Q}; \psi; \sigma; i)$$

  We have that $\mathcal{Q} = \mathcal{Q}_S$, $\psi = \psi_S\lambda$, $i = i_S$, and $\sigma = \sigma^- \uplus \tau\lambda = \sigma_S^-\lambda \uplus \tau\lambda = (\sigma_S^- \uplus \tau)\lambda = \sigma_S\lambda$. It gives us the result.
- *Case* $\alpha_S = \mathsf{out}(c, \mathsf{w})$. In such a case, we have that $\mathcal{Q}_S^- = \mathcal{Q}^- = \{\mathsf{out}(c, u).Q_c\} \uplus \mathcal{P}$, and we have that $\psi_S = \psi_S^- \uplus \{\mathsf{w} \triangleright u\sigma_S^-\}$. Let $\mathcal{Q} = \{Q_c\} \uplus \mathcal{P}$, $\psi = \psi^- \uplus \{\mathsf{w} \triangleright u\sigma^-\}$, $\sigma = \sigma^-$, and $i = i^-$. We now show that $u\sigma^- = u\sigma_S^-\lambda$ is a Σ_0^--message. We know that $u\sigma_S^-$ is a Σ_0^+-message. Moreover, thanks to Lemma 4.8, we know that $c\lambda$ is a Σ_0^--message for each $c \in dom(\lambda)$, and $c\lambda$ is atomic when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$. This allows us to conclude that $u\sigma^- = u\sigma_S^-\lambda$ is a Σ_0^--message. Therefore, we have that:

$$(\{\mathsf{out}(c, u).Q_c\} \uplus \mathcal{P}; \psi^-; \sigma^-; i^-) \xrightarrow{\mathsf{out}(c, \mathsf{w})} (\{Q_c\} \uplus \mathcal{P}; \psi^- \uplus \{\mathsf{w} \triangleright u\sigma^-\}; \sigma^-; i^-) = (\mathcal{Q}; \psi; \sigma; i).$$

  We have that $\mathcal{Q} = \mathcal{Q}_S$, $\sigma = \sigma_S\lambda$, $i = i_S$, and $\psi = \psi_S\lambda$ since $\sigma_S^-\lambda = \sigma^-$ by induction hypothesis. It gives us the result.

We may note that the resulting trace $\mathsf{tr}_S\theta$ only contains Σ_0^--recipes: constants from Σ_{fresh} occurring in tr_S have been replaced by θ. Hence, the result. □

Proposition 5.4. *Let* $\mathcal{K}_P$ *and* $\mathcal{K}_Q$ *be two initial* Σ_0^-*-configurations such that* $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$ *w.r.t.* Σ_0^-. *Let* $(\mathsf{tr}, \phi) \in \mathsf{trace}_{\Sigma_0^-}(\mathcal{K}_P)$ *with underlying substitution* σ *be a witness of non-inclusion. Then, there exists* $(\mathsf{tr}_S, \phi_S) \in \mathsf{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ *a witness of this non-inclusion with underlying substitution* σ_S *such that* $\sigma_S = mgu(\Gamma)\rho$, *as well as two substitutions* λ_P *and* θ *such that* $(\rho, \lambda_P, \theta)$ *is a well-typed concretization of* (tr_S, ϕ_S) *w.r.t.* (tr, ϕ) *in the context* $\mathcal{K}_P$, *as defined in Proposition 4.11.*

Proof. Let $(\mathsf{tr}, \phi) \in \mathsf{trace}_{\Sigma_0^-}(\mathcal{K}_P)$ be a witness of non-inclusion with underlying substitution σ. First, we apply Proposition 4.11. We deduce that there exists $(\mathsf{tr}_S, \phi_S) \in \mathsf{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ with underlying substitution σ_S with $dom(\sigma_S) = dom(\sigma)$ such that $\sigma_S = mgu(\Gamma)\rho$, as well as two substitutions λ_P and θ such that:

- $\Gamma = \{(u, v) \mid u, v \in ESt(\mathcal{K}_P)$ such that $u\sigma = v\sigma\}$.
- ρ is a bijective renaming from variables in $dom(\sigma) \setminus dom(mgu(\Gamma))$ to constants in Σ_{fresh} such that $x\rho \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$ if, and only if, $x\sigma$ is an atomic Σ_0^--message.
- $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$, for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in tr_S, we have that $c \in dom(\theta), c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S))$, and $\mathsf{w}' \prec c$ for any $\mathsf{w}' \in vars(c\theta)$ (where $\prec$ is the ordering induced by tr_S).
- for any $c \in dom(\theta)$, $(c\theta)\phi_S\downarrow$ is a Σ_0^+-message and it is an atom when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$.
- λ_P is the first-order substitution associated to θ through ϕ_S.
- $\phi = \phi_S\lambda_P$, $\sigma = \sigma_S\lambda_P$, and $(\mathsf{tr}_S\theta)\phi\downarrow = \mathsf{tr}\phi\downarrow$.

Our goal is to show that $(\mathrm{tr}_S, \phi_S) \in \mathrm{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ is a witness of $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$ w.r.t Σ_0^+.

Let ψ_S be such that $(\mathrm{tr}_S, \psi_S) \in \mathrm{trace}_{\Sigma_0^+}(\mathcal{K}_Q)$ and $\phi_S \sqsubseteq_s \psi_S$ (and thus $dom(\phi_S) = dom(\psi_S)$). Note that such a ψ_S exists since otherwise the result trivially holds.

We have that ϕ_S is a Σ_0^+-frame and $\prec$ is an ordering on $dom(\phi_S)$. We have $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$ and $c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S))$ for any $c \in dom(\theta)$. Moreover, if $c \in \Sigma_{\mathsf{fresh}}$ occurs in $\mathrm{w}\phi_S$, then, as c does not occur in $\mathcal{K}_P$, it must have been introduced in an input before the output of w. So $c \prec \mathrm{w}$. Let $\mathrm{w}' \in vars(c\theta)$. Then $\mathrm{w}' \prec c$ by definition of $\prec$. So $\mathrm{w}' \prec \mathrm{w}$ and the order induced by $\prec$ on $dom(\phi_S)$ satisfies the condition of Lemma 4.8 and Lemma 4.9. $(c\theta)\phi_S\!\downarrow$ is a Σ_0^+-message for each $c \in dom(\theta)$ and $(c\theta)\phi_S\!\downarrow$ is an atomic Σ_0^+-message when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$. So Lemma 4.8 applies and for each $c \in dom(\lambda_P)$, $c\lambda_P$ is a Σ_0^--message and $c\lambda_P$ is atomic if $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$. So Lemma 4.9 applies, and for any recipe R such that $R\phi_S\!\downarrow$ is a Σ_0^+-message, we have $(R\theta)(\phi_S\lambda_P)\!\downarrow = (R\phi_S\!\downarrow)\lambda_P$.

$\mathcal{K}_Q$ is a Σ_0^--configuration, and $(\mathrm{tr}_S, \psi_S) \in \mathrm{trace}_{\Sigma_0^+}(\mathcal{K}_Q)$. Moreover, $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$, and $c \in dom(\theta)$ for each c occurring in tr_S. We have $dom(\phi_S) = dom(\psi_S)$ by $\phi_S \sqsubseteq_s \psi_S$ so for each $c \in dom(\theta)$, $c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S)) \subseteq \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\psi_S))$. Moreover, if $c \in \Sigma_{\mathsf{fresh}}$ occurs in $\mathrm{w}\psi_S$, then, as c does not occur in the protocol, it must have been introduced in an input before the output of w. So $c \prec \mathrm{w}$. Let $\mathrm{w}' \in vars(c\theta)$. Then $\mathrm{w}' \prec c$ by definition of $\prec$. So $\mathrm{w}' \prec \mathrm{w}$ and the order induced by $\prec$ on $dom(\psi_S)$ satisfies the condition of Lemma 4.8 and Lemma 4.9. By $\phi_S \sqsubseteq_s \psi_S$, $(c\theta)\psi_S\!\downarrow$ is a Σ_0^+-message for any $c \in dom(\theta)$ and an atomic one in case $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$. Hence, we have that both Lemma B.1 and Lemma 4.8 apply. We obtain that $(\mathrm{tr}_S\theta, \psi_S\lambda_Q) \in \mathrm{trace}_{\Sigma_0^-}(\mathcal{K}_Q)$ where λ_Q is the first-order substitution associated to θ through ψ_S. We also get that $c\lambda_Q$ is a Σ_0^--message for each $c \in dom(\lambda_Q)$ and it is atomic when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$. Therefore, Lemma 4.9 applies, and we have that for any $R \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\theta) \uplus dom(\psi_S))$ such that $R\psi_S\!\downarrow$ is a Σ_0^+-message, $(R\theta)(\psi_S\lambda_Q)\!\downarrow = (R\psi_S\!\downarrow)\lambda_Q$. Moreover, since λ_Q preserves atomicity, $(R\psi_S\!\downarrow)\lambda_Q$ is a Σ_0^--message whenever $(R\psi_S\!\downarrow)$ is a Σ_0^+-message.

In the remaining of this proof, we suppose that $\phi_S \sqsubseteq_s \psi_S$ (meaning that $(\mathrm{tr}_S, \phi_S) \in \mathrm{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ is not a witness), and we show that $\phi \sqsubseteq_s \psi$ leading to a contradiction since by hypothesis, we know that $(\mathrm{tr}, \phi) \in \mathrm{trace}_{\Sigma_0^-}(\mathcal{K}_P)$ is a witness, and tr passes in $\mathcal{K}_Q$ leading to the frame ψ.

To establish this result, we rely on Lemma 5.3 and thus we reason with the notion $\sqsubseteq_s^{\mathsf{atom}}$. We consider a test T (built on $\mathcal{T}(\Sigma, \Sigma_0^+ \cup dom(\phi_S))$ such that $T\theta$ holds in ϕ. We assume that for all T' such that $\mu_{\phi_S}(T') < \mu_{\phi_S}(T)$, we have that:

$$T'\theta \text{ holds in } \phi \text{ implies that } T'\theta \text{ holds in } \psi.$$

We have to prove that $T\theta$ holds in ψ. We distinguish three cases depending on the form of the test:

(1) The test T is a Σ_0^+-recipe R such that $(R\theta)\phi\!\downarrow$ is a Σ_0^--message. In such a case, we have to establish that $(R\theta)\psi\!\downarrow$ is a Σ_0^--message
(2) The test T is a Σ_0^+-recipe such that $(R\theta)\phi\!\downarrow$ is an atomic Σ_0^--message, i.e. $(R\theta)\phi\!\downarrow \in \Sigma_0^- \uplus \mathcal{N}$. In such a case, we have to establish that $(R\theta)\psi\!\downarrow$ is an atomic Σ_0^--message.
(3) The test T is made of two Σ_0^+-recipes R, R' such that both $(R\theta)\phi\!\downarrow$ and $(R'\theta)\phi\!\downarrow$ are Σ_0^--messages, and $(R\theta)\phi\!\downarrow = (R'\theta)\phi\!\downarrow$. In such a case, we have to establish that $(R\theta)\psi\!\downarrow = (R'\theta)\psi\!\downarrow$.

(1) *R is Σ_0^+-recipe such that $(R\theta)\phi\!\downarrow$ is a Σ_0^--message.*

Assume that $R\phi_S\!\downarrow$ is not a Σ_0^+-message. We take the smallest subterm R' of R such that $R'\phi_S\!\downarrow$ is not a Σ_0^+-message. Let p be such that $R|_p = R'$. As $R'\phi_S\!\downarrow$ is not a Σ_0^+-message, we know that $R' \notin \Sigma_0^+ \uplus dom(\phi_S)$. Thus, we have that $R' = \mathrm{f}(R_1, \ldots, R_k)$ for some $\mathrm{f} \in \Sigma$. Moreover, by minimality of R', we know that $R_i\phi_S\!\downarrow$ is a Σ_0^+-message for $1 \leq i \leq k$. We now establish the following claim:

Claim. If $R_i\phi_S\!\downarrow \in \Sigma_{\mathsf{fresh}}$ for some $i \in \{1, \ldots, k\}$, then $(R\theta)\psi\!\downarrow$ is a Σ_0^--message.

Proof. Assume $R_i\phi_S\downarrow = c \in \Sigma_{\mathsf{fresh}}$ for some $i \in \{1, \ldots, k\}$, and let $R'_i = c\theta$. We have that $R'_i\theta = c\theta$, and thus $(R'_i\theta)\phi\downarrow = (c\theta)\phi\downarrow$, and $(R'_i\theta)\psi\downarrow = (c\theta)\psi\downarrow$. We have that the test $c = R_i$ holds in ϕ_S, and thus since $\phi_S \sqsubseteq_s \psi_S$, it also holds in ψ_S. Lemma 4.9 applies, and we obtain: $(R_i\theta)\psi\downarrow = (R_i\psi_S\downarrow)\lambda_Q = c\lambda_Q = (c\theta)\psi\downarrow = (R'_i\theta)\psi\downarrow$. Let $\overline{R} = R[R'_i]_{(p.i)}$. We have that:

- $(\overline{R}\theta = R\theta)$ holds in both ϕ and ψ;
- $\mu^1_{\phi_S}(R'_i) < \mu^1_{\phi_S}(R_i)$ thanks to Lemma A.4, and thus $\mu^1_{\phi_S}(\overline{R}) < \mu^1_{\phi_S}(R)$ thanks to Lemma A.3 and Lemma 4.6.

As $(\overline{R}\theta)\phi\downarrow = (R\theta)\phi\downarrow$ is a Σ_0^--message, such a test transfers to ψ (relying on our induction hypothesis), and $(\overline{R}\theta)\psi\downarrow$ is a Σ_0^--message. It proves our claim.

We now distinguish two cases depending on whether $\mathsf{f} \in \Sigma_{\mathsf{c}}$ or $\mathsf{f} \in \Sigma_{\mathsf{d}}$.

Case where $\mathsf{f} \in \Sigma_{\mathsf{c}}$. In such a case, $R'\phi_S\downarrow$ is either not well-shaped or not well-sorted. If it is not well-sorted, then for one of its subrecipes R_i, $R_i\phi_S\downarrow$ is not an atom (it is well-sorted by minimality of R') while it should be. In particular, $(R_i\theta)\phi\downarrow = R_i\phi_S\downarrow\lambda_P$ is an atom. So $R_i\phi_S\downarrow$ must be in Σ_{fresh}, which implies by our claim that $(R\theta)\psi\downarrow$ is a Σ_0^--message, and thus we are done. We deduce that $R'\phi_S\downarrow$ is well-sorted.

Now, we assume that $R'\phi_S\downarrow$ is not well-shaped, we consider the shape of f, $sh_{\mathsf{f}} = \mathsf{f}(s_1, \ldots, s_k)$ for some $s_1, \ldots, s_k$. As $R'\phi_S\downarrow$ has a bad shape and is a f-term, there must be a j such that $R_j\phi_S\downarrow$ is not an instance of s_j. In particular, s_j is not a variable and we have $s_j = sh_{\mathsf{g}}$ for some function symbol g (thanks to the compatibility of the shapes). But $R_j\phi_S\downarrow$ is a Σ_0^+-message and thus we know that $(R_j\theta)\phi\downarrow = (R_j\phi_S\downarrow)\lambda_P$ (thanks to Lemma 4.9) is an instance of s_j as $(R_j\theta)\phi\downarrow$ is a Σ_0^+-message. Relying on our claim, we know that $R_j\phi_S\downarrow \notin \Sigma_{\mathsf{fresh}}$, thus $R_j\phi_S\downarrow$ is a g-term, and since we know that $R_j\phi_S\downarrow$ is a Σ_0^+-message, we know that it is an instance of s_j, yielding to a contradiction.

Case where $\mathsf{f} = \mathsf{des} \in \Sigma_{\mathsf{d}}$. In such a case, $R' = \mathsf{des}(R_1, \ldots, R_k)$. Let $\ell_{\mathsf{des}} = \mathsf{des}(t_1, t_2, \ldots, t_k)$. We distinguish two subcases.

First, we assume that there is some $i \in \{1, \ldots, k\}$ such that $R_i\phi_S\downarrow$ does not unify with t_i. As $R_i\phi_S\downarrow$ is a Σ_0^+-message, it has a good shape. Thus, the only way to not unify with the linear term t_i whereas $(R_i\phi_S)\lambda_P = (R_i\theta)\phi\downarrow$ (thanks to Lemma 4.9) does, is when $R_i\phi_S\downarrow = c$ for some $c \in \Sigma_{\mathsf{fresh}}$. Relying on our claim, we obtain a contradiction.

Second, we assume that $R_i\phi_S\downarrow$ unifies with t_i for each i. In such a case, we know that $\ell_{\mathsf{des}} = \mathsf{des}(t_1, \ldots, t_k)$ is a non-linear term and we denote x the non-linear variable occurring in ℓ_{des}. Let $I_0 = \{1 \leq i \leq k \mid x \text{ occurs in } t_i\}$. We know that $1 \in I_0$. For any $i \in I_0$, we denote p_i the position in t_i such that $t_i|_{p_i} = x$. Since $R'\phi_S\downarrow\lambda_P$ is a Σ_0^--message, we know that $t = \mathsf{des}(R_1\phi_S\downarrow\lambda_P, \ldots, R_k\phi_S\downarrow\lambda_P)$ unifies with $\mathsf{des}(t_1, \ldots, t_k)$. Therefore, we know that there exists an atomic Σ_0^--message a such that $t|_{i.p_i} = a$ for any $i \in I_0$. We know that $R'\phi_S\downarrow$ is not a Σ_0^+-message whereas $R_i\phi_S\downarrow$ are Σ_0^+-messages. Thus, we have that $t_S = \mathsf{des}(R_1\phi_S\downarrow, \ldots, R_k\phi_S\downarrow)$ does not unify with $\mathsf{des}(t_1, \ldots, t_k)$. We deduce the following fact:

Fact: for any $i \in I_0$, we have $t_S|_{i.p_i} = a$, or $t_S|_{i.p_i} = c$ for some $c \in \Sigma_{\mathsf{fresh}}$ such that $c\lambda_P = (c\theta)\phi\downarrow = a$.

Let c be a constant from Σ_{fresh} such that $t_S|_{i.p_i} = c$ for some $i \in I_0$. Let $I_1 = \{i \in I_0 \mid R_i\phi_S\downarrow|_{p_i} = c\}$. We will build two recipes $\overline{R}'$ and $\overline{R}''$ derived from R' and enjoying some nice properties: in particular both $(\overline{R}'\theta)\psi\downarrow$ and $(\overline{R}''\theta)\psi\downarrow$ will be Σ_0^--message, and this will allow us to derive that $(R\theta)\psi\downarrow$ is a Σ_0^--message too.

Construction of $\overline{R'}$. Let ν'_{des} be the substitution such that $x\nu'_{\mathsf{des}} = c\theta$, and $y\nu'_{\mathsf{des}} = c_{\mathsf{min}}$ for any other variable $y \in vars(\ell_{\mathsf{des}})$. For $i \in \{1, \ldots, k\}$, let $R'_i = t_i\nu'_{\mathsf{des}}$ in case $i \in I_1$, and $R'_i = R_i$ otherwise. Let $\overline{R}' = \mathsf{des}(R'_1, \ldots, R'_k)$. Each $R'_i\phi_S\downarrow$ is a Σ_0^+-message, and thus by Lemma 4.9, we have

that $(R'_i\theta)\phi\downarrow = R'_i\phi_S\downarrow\lambda_P$ is a Σ_0^--message. By construction of $\overline{R}'$ and relying on our fact, we have that $(\overline{R}'\theta)\phi\downarrow$ is a Σ_0^--message. Therefore, relying on Lemma A.3, and since $\mu^1_{\phi_S}(R'_i) < \mu^1_{\phi_S}(R_i)$ for $i \in I_1 \neq \emptyset$, and $\mu^1_{\phi_S}(R'_i) = \mu^1_{\phi_S}(R_i)$ otherwise, we deduce that:

$$\begin{array}{rcl}
\mu^1_{\phi_S^-}(\overline{R}') & = & \mathsf{Multi}(\overline{R'}\phi_S^-\downarrow) \\
& \leq & \mathsf{Multi}(\mathsf{des}(R'_1\phi_S\downarrow, \ldots, R'_k\phi_S\downarrow)) \\
& = & \{\mathsf{des}\} \uplus \mu^1_{\phi_S}(R'_1) \uplus \cdots \uplus \mu^1_{\phi_S}(R'_k) \\
& < & \{\mathsf{des}\} \uplus \mu^1_{\phi_S}(R_1) \uplus \cdots \uplus \mu^1_{\phi_S}(R_k) \\
& = & \mu^1_{\phi_S}(R)
\end{array}$$

As $(\overline{R}'\theta)\phi\downarrow$ is a Σ_0^--message, we deduce that $(\overline{R}'\theta)\psi\downarrow$ is a Σ_0^--message. Since $R'\phi_S\downarrow$ is not a Σ_0^+-message, Lemma 4.6 applies, and we obtain $\mu^1_{\phi_S}(R[\overline{R}'\,]_p) < \mu^1_{\phi_S}(R[R'\,]_p) = \mu^1_{\phi_S}(R)$.

Construction of $\overline{R''}$. Let ν''_{des} be the substitution such that $x\nu''_{\mathsf{des}} = c$, and $y\nu''_{\mathsf{des}} = c_{\mathsf{min}}$ for any other variable $y \in vars(\ell_{\mathsf{des}})$. For $i \in \{1, \ldots, k\}$, let $R''_i = R_i$ in case $i \in I_1$, and $R''_i = t_i\nu''_{\mathsf{des}}$ otherwise. Let $\overline{R}'' = \mathsf{des}(R''_1, \ldots, R''_k)$. By construction of $\overline{R}''$, we have that $\overline{R}''\phi_S\downarrow$ is a Σ_0^+-message. By hypothesis we have that $\phi_S \sqsubseteq_s \psi_S$, and thus we deduce that $\overline{R}''\psi_S\downarrow$ is a Σ_0^+-message. Then, thanks to Lemma 4.9, we deduce that $(\overline{R}''\theta)\psi\downarrow = \overline{R}''\psi_S\downarrow\lambda_Q$. Regarding the measure, we have that $\{\mathsf{des}\} < \mu^1_{\phi_S}(R')$ since des occurs in $R'\phi_S\downarrow$. We have that $\mu^1_{\phi_S}(\overline{R}'') < \{\mathsf{des}\}$ since $\overline{R}''\phi_S\downarrow$ is a Σ_0^+-message. Therefore, we have that $\mu^1_{\phi_S}(\overline{R}'') < \mu^1_{\phi_S}(R')$. Since $R'\phi_S\downarrow$ is not a Σ_0^+-message, Lemma 4.6 applies, and we obtain $\mu^1_{\phi_S}(R[\overline{R}''\,]_p) < \mu^1_{\phi_S}(R[R'\,]_p) = \mu^1_{\phi_S}(R)$.

At this point, we have that $\overline{R}' = \mathsf{des}(R'_1, \ldots, R'_k)$, $\overline{R}'' = \mathsf{des}(R''_1, \ldots, R''_k)$, and both $(\overline{R}'\theta)\psi\downarrow$ and $(\overline{R}''\theta)\psi\downarrow$ are Σ_0^--message. By construction, for each $1 \leq i \leq k$, we have that either $R_i = R'_i$ or $R_i = R''_i$. Therefore, for each $1 \leq i \leq k$, we have that $(R_i\theta)\psi\downarrow$ is a Σ_0^--message and unifies with t_i. We put a $c\theta$ in atomic positions of ℓ_{des} in some R'_i. As $(\overline{R}'\theta)\psi\downarrow$ is a Σ_0^- message, it means that each of the $(R'_i\theta)\psi\downarrow$ has $(c\theta)\psi\downarrow$ in atomic position of ℓ_{des}. Similarly, we get that each of the $(R''_i\theta)\psi\downarrow$ has $(c\theta)\psi\downarrow$ in atomic position of ℓ_{des}. So there is $(c\theta)\psi\downarrow$ in atomic position of ℓ_{des} for each $(R_i\theta)\psi\downarrow$. We deduce that $(R'\theta)\psi\downarrow$ is a Σ_0^--message.

Furthermore, we have $R_1 = R'_1$ or $R_1 = R''_1$. Let $\overline{R} = \overline{R}'$ in case $R_1 = R'_1$, and $\overline{R} = \overline{R}''$ in case $R_1 = R''_1$. We have that:

- $\mu^1_{\phi_S}(R[\overline{R}]_p) < \mu^1_{\phi_S}(R)$,
- $(R[\overline{R}]_p\theta)\phi\downarrow = (R[R'\,]_p\theta)\phi\downarrow = (R\theta)\phi\downarrow$, and
- $(R[\overline{R}]_p\theta)\psi\downarrow = (R\theta)\psi\downarrow$.

As $(R[\overline{R}]_p\theta)\phi\downarrow$ is a Σ_0^--message, by minimality $(R[\overline{R}]_p\theta)\psi\downarrow$ is a Σ_0^--message, and so $(R\theta)\psi\downarrow$ is a Σ_0^--message which is the result we want to prove.

Thus, we have that $R\phi_S\downarrow$ is a Σ_0^+-message. By $\phi_S \sqsubseteq_s \psi_S$, we know that $R\psi_S\downarrow$ is a Σ_0^+-message, and Lemma 4.9 allows us to conclude that $(R\theta)\psi\downarrow = R\psi_S\downarrow\lambda_Q$ is a Σ_0^--message.

(2) *R is a Σ_0^+-recipe such that $(R\theta)\phi\downarrow$ is an atomic Σ_0^--message.*

First, we have that $R\phi_S\downarrow$ is a Σ_0^+-message (see case (1)), and $(R\theta)\phi\downarrow = R\phi_S\downarrow\lambda_P$ thanks to Lemma 4.9. As a first step, we establish that $R\psi_S\downarrow$ is atomic. As $R\phi_S\downarrow\lambda_P$ is an atom, we know that $R\phi_S\downarrow$ is either an atom from $\Sigma_0^- \cup \mathcal{N}$, or a constant from Σ_{fresh}.

- If $R\phi_S\downarrow \notin \Sigma_{\mathsf{fresh}}$, then $R\phi_S\downarrow = R\phi_S\downarrow\lambda_P$ is atomic, and relying on our hypothesis $\phi_S \sqsubseteq_s \psi_S$, we deduce that $R\psi_S\downarrow$ is atomic.
- If $R\phi_S\downarrow = c \in \Sigma_{\mathsf{fresh}}$, then there exists x such that $x\rho = c$. Since $x \in dom(\rho)$, we know that $x \notin dom(mgu(\Gamma))$, and thus $xmgu(\Gamma) = x$. Therefore, we have that $x\sigma = x\sigma_S\lambda_P = x(mgu(\Gamma)\rho)\lambda_P = (x\rho)\lambda_P = c\lambda_P = R\phi_S\downarrow\lambda_P$. As $R\phi_S\downarrow\lambda_P$ is an atom, we deduce that $x\sigma$ is atomic, and by definition of ρ, we have that $c = x\rho \in \Sigma^{\mathsf{atom}}_{\mathsf{fresh}}$, and thus $R\phi_S\downarrow$ is atomic. Relying on our hypothesis $\phi_S \sqsubseteq_s \psi_S$, we deduce that $R\psi_S\downarrow = c$ is atomic.

Since λ_Q replaces atoms by atoms, we deduce in both cases that $R\psi_S\downarrow\lambda_Q$ is atomic, and thus $(R\theta)\psi\downarrow = (R\psi_S\downarrow)\lambda_Q$ (Lemma 4.9) is an atomic Σ_0^--message.

(3) *R and R' are Σ_0^+-recipes, $(R\theta)\phi\downarrow$, $(R'\theta)\phi\downarrow$ are Σ_0^--messages, and $(R\theta)\phi\downarrow = (R'\theta)\phi\downarrow$.*

Step 1: We prove that R and R' are simple Σ_0^+-recipe, in normal form w.r.t. $\downarrow$. Moreover, we show that R' is a subterm recipe such that $R'\phi_S\downarrow$ is either an encrypted term, or a name from $\mathcal{N}$, or a constant from Σ_0^+. Regarding R, we show that R is of the form $C[R_1, \ldots, R_n]$ where for each $i \in \{1, \ldots, n\}$, we have that $R_i\phi_S\downarrow$ is either an encrypted term, or a name from $\mathcal{N}$, or a constant from Σ_0^+.

We have that $(R\theta)\phi\downarrow$ and $(R'\theta)\phi\downarrow$ are Σ_0^--messages, $\mu_{\phi_S}(R) < \mu_{\phi_S}(R = R')$ and $\mu_{\phi_S}(R') < \mu_{\phi_S}(R = R')$. Hence, we deduce that $(R\theta)\psi\downarrow$ and $(R'\theta)\psi\downarrow$ are Σ_0^--messages.

Assume that either R or R' is not in normal form w.r.t. $\downarrow$, say R. Then $\mu^1_{\phi_S}(R\downarrow) < \mu^1_{\phi_S}(R)$ by Lemma 4.7. Moreover, as $R\theta \twoheadrightarrow^* R\downarrow\theta$, Lemma 4.3 applies: $(R\downarrow\theta)\phi\downarrow = (R\theta)\phi\downarrow$ and $(R\downarrow\theta)\psi\downarrow = (R\theta)\psi\downarrow$ as $(R\theta)$ gives a Σ_0^--message in both ϕ and ψ. So $(R\downarrow\theta = R'\theta)$ holds in ϕ. By $\mu_{\phi_S}(R\downarrow = R') < \mu_{\phi_S}(R = R')$, it transfers to ψ and we deduce $(R\theta)\psi\downarrow = (R\downarrow\theta)\psi\downarrow = (R'\theta)\psi\downarrow$ which is the result we want to establish.

Now, we assume that both R and R' are in normal form w.r.t. $\downarrow$. Thus, both are simple by Lemma 4.5. In case both R and R' have a constructor as root symbol, then this is necessarily the same. Therefore, we have that $R = \mathsf{f}(R_1, \ldots, R_k)$ and $R' = \mathsf{f}(R'_1, \ldots, R'_k)$. We have that $(R_i\theta)\phi\downarrow$ and $(R'_i\theta)\phi\downarrow$ are Σ_0^+-messages $(1 \le i \le k)$, and $(R_i\theta)\phi\downarrow = (R'_i\theta)\phi\downarrow$ $(1 \le i \le k)$. Since $\mu_{\phi_S}(R_i = R'_i) < \mu_{\phi_S}(R = R')$, we deduce that $(R_i\theta)\psi\downarrow = (R'_i\theta)\psi\downarrow$, and thus $(R\theta)\psi\downarrow = (R'\theta)\psi\downarrow$ which is the result we want to prove.

Therefore we can assume that say R' is a subterm-recipe, and R is simple, so $R = C[R_1, \ldots, R_n]$ where for each i, R_i is a subterm-recipe. Now, assume that there is i_0 such that $\mathrm{root}(R_{i_0}\phi_S\downarrow) = \mathsf{f} \in \Sigma_{\mathsf{c}}$ and f is transparent, then $R_{i_0}\phi_S\downarrow = \mathsf{f}(C^{\mathsf{f}}_1[R_{i_0}], \ldots, C^{\mathsf{f}}_k[R_{i_0}])\phi_S\downarrow$. We consider the context $\overline{C}$ such that $\overline{C}[R_1, \ldots, R_n] = C[R_1, \ldots, \mathsf{f}(C^{\mathsf{f}}_1[R_{i_0}], \ldots, C^{\mathsf{f}}_k[R_{i_0}]), \ldots, R_n]$. We have that $\overline{R} = \overline{C}[R_1, \ldots, R_n]$ is a Σ_0^+-recipe such that $\overline{R}\phi_S\downarrow = R\phi_S\downarrow$, and thus $\mu^1_{\phi_S}(R) = \mu^1_{\phi_S}(\overline{R})$, and $(\overline{R}\theta)\phi\downarrow = \overline{R}\phi_S\downarrow\lambda_P$ by Lemma 4.9, which gives $(\overline{R}\theta)\phi\downarrow = \overline{R}\phi_S\downarrow\lambda_P = R\phi_S\downarrow\lambda_P = (R\theta)\phi\downarrow$. We have that $\mu^2_{\phi_S}(\overline{R}) < \mu^2_{\phi_S}(R)$. The equality $\overline{R}\phi_S\downarrow = R\phi_S\downarrow$ transfers to $\overline{R}\psi_S\downarrow = R\psi_S\downarrow$ by $\phi_S \sqsubseteq_s \psi_S$ and we deduce $(\overline{R}\theta)\psi\downarrow = (R\theta)\psi\downarrow$ by Lemma 4.9. As $\mu_{\phi_S}(\overline{R} = R') < \mu_{\phi_S}(R = R')$, the equality $(\overline{R}\theta)\phi\downarrow = (R'\theta)\phi\downarrow$ transfers to $(\overline{R}\theta)\psi\downarrow = (R'\theta)\psi\downarrow$ and we deduce that $(R\theta)\psi\downarrow = (R'\theta)\psi\downarrow$ which is the result we want to prove.

Therefore, we deduce that each $R_i\phi_S\downarrow$ (with $1 \le i \le n$) is either an encrypted term, a constant from Σ_0^+, or a name from $\mathcal{N}$. A similar reasoning allows us to establish that $R'\phi_S\downarrow$ is either an encrypted term, a constant from Σ_0^+, or a name from $\mathcal{N}$.

Step 2: We now establish that $(R\theta)\psi\downarrow = (R'\theta)\psi\downarrow$.

Let $t = R\phi_S\downarrow$ and $v = R'\phi_S\downarrow$. By Lemma 4.9 and $(R\theta)\phi\downarrow = (R'\theta)\phi\downarrow$, we know that $t\lambda_P = v\lambda_P$. If $t = v$, then we have that $R\phi_S\downarrow = R'\phi_S\downarrow$ since $\phi_S \sqsubseteq_s \psi_S$. Then by Lemma 4.9, we deduce that $(R\theta)\psi\downarrow = (R'\theta)\psi\downarrow$.

From now on, we assume that $t \neq v$. Since $t \neq v$, we know that there exists a position p defined in t and v such that $\text{root}(t|_p) \neq \text{root}(v|_p)$. Let p be any position defined in t and v such that $\text{root}(t|_p) \neq \text{root}(v|_p)$. Since $t\lambda_P = v\lambda_P$, and $dom(\lambda_P) \subseteq \Sigma_{\text{fresh}}$, we have that $t|_p \in \Sigma_{\text{fresh}}$ or $v|_p \in \Sigma_{\text{fresh}}$.

We first assume that there exists such a position p that falls outside the context C. More precisely, we have that $p = p'.p''$ (with p' a strict prefix of p) and $C[R_1, \ldots, R_n]|_{p'} = R_{i_0}$ for some $i_0 \in \{1, \ldots, n\}$. Therefore, since $R_{i_0}\phi_S\downarrow$ is not a leaf ($p'' \neq \epsilon$), we know that $t|_{p'} = R_{i_0}\phi_S\downarrow$ is an encrypted subterm of ϕ_S. Relying on Lemma 4.10, we have that:

- $t|_{p'} \in ESt(\phi_S) \subseteq ESt(\mathcal{K}_S\sigma_S) \subseteq ESt(\mathcal{K}_P(mgu(\Gamma)\rho)) \subseteq ESt(\mathcal{K}_P)\sigma_S$.
- $v|_{p'} \in ESt(\phi_S) \subseteq ESt(\mathcal{K}_S\sigma_S) \subseteq ESt(\mathcal{K}_P(mgu(\Gamma)\rho)) \subseteq ESt(\mathcal{K}_P)\sigma_S$.

This allows us to conclude that there exist $t', v' \in ESt(\mathcal{K}_P)$ such that $t'\sigma_S = t|_{p'}$, and $v'\sigma_S = v|_{p'}$. Since $t\lambda_P = v\lambda_P$, we know that $(t\lambda_P)|_{p'} = (v\lambda_P)|_{p'}$, thus $t|_{p'}\lambda_P = v|_{p'}\lambda_P$, and $(t'\sigma_S)\lambda_P = (v'\sigma_S)\lambda_P$. We have that $(t'\sigma_S)\lambda_P = t'(\sigma_S\lambda_P)$ and also that $(v'\sigma_S)\lambda_P = v'(\sigma_S\lambda_P)$. Since we have that $\sigma = \sigma_S\lambda_P$, we deduce that $t'\sigma = v'\sigma$. By definition of σ_S, we have that $t'\sigma_S = v'\sigma_S$. Thus, we have that $t|_{p'} = v|_{p'}$ leading to a contradiction since we have assumed that t and v differ below the position p'.

Thus, we know that for any position p defined in t and v such that $\text{root}(t|_p) \neq \text{root}(v|_p)$, we have that $t|_p$ or $v|_p$ is in Σ_{fresh}, and p is a position of C.

If $t|_p = c \in \Sigma_{\text{fresh}}$, let $\overline{R} = R[c\theta]_p$. We have that $\mu^1_{\phi_S}(\overline{R}) < \mu^1_{\phi_S}(R)$ since $\mu^1_{\phi_S}(c\theta) < \mu^1_{\phi_S}(R|_p)$ (note that $R|_p\phi_S\downarrow = c$ whereas $(c\theta)\phi_S\downarrow$ is a Σ_0^+-message and it is an atom when $c \in \Sigma^{\text{atom}}_{\text{fresh}}$). Moreover, we have that:

- $(\overline{R}\theta)\phi\downarrow = (R\theta)\phi\downarrow = (R'\theta)\phi\downarrow$, and
- $\mu_{\phi_S}(\overline{R} = R') < \mu_{\phi_S}(R = R')$.

Thus, by induction hypothesis, we have that $(\overline{R}\theta)\psi\downarrow = (R'\theta)\psi\downarrow$. We also have $R|_p\phi_S\downarrow = c$ so by $\phi_S \sqsubseteq_s \psi_S$, we know that $R|_p\psi_S\downarrow = c$. By Lemma 4.9, $(R|_p\theta)\psi\downarrow = (c\theta)\psi\downarrow$. So $(\overline{R}\theta)\psi\downarrow = (R\theta)\psi\downarrow$. We deduce $(R\theta)\psi\downarrow = (R'\theta)\psi\downarrow$ which is the result we want to prove.

Hence, from now on, we assume that $t|_p \notin \Sigma_{\text{fresh}}$, and thus $v|_p = c$ for some $c \in \Sigma_{\text{fresh}}$. Let $p_1, \ldots, p_m$ be the positions such that for each $i \in \{1, \ldots, m\}$, we have that p_i is defined in both t and v, and $\text{root}(t|_{p_i}) \neq \text{root}(v|_{p_i})$. For each $i \in \{1, \ldots, m\}$, we know that p_i is a position of C such that $t|_{p_i} \notin \Sigma_{\text{fresh}}$, and $v|_{p_i} \in \Sigma_{\text{fresh}}$. We denote c^i_{fresh} the constant from Σ_{fresh} such that $v|_{p_i} = c^i_{\text{fresh}}$. We have $(c^i_{\text{fresh}}\theta)\phi\downarrow = (R|_{p_i}\theta)\phi\downarrow$. $\mu^1_{\phi_S}(c^i_{\text{fresh}}) < \mu^1_{\phi_S}(R')$ and $\mu^1_{\phi_S}(R|_{p_i}) \leq \mu^1_{\phi_S}(R)$ so $\mu_{\phi_S}(c^i_{\text{fresh}} = R|_{p_i}) < \mu_{\phi_S}(R = R')$. We deduce $(c^i_{\text{fresh}}\theta)\psi\downarrow = (R|_{p_i}\theta)\psi\downarrow$.

Now, let $\overline{C}$ be the context obtained from C by putting c^i_{fresh} at position p_i for each i. Let $\overline{R} = \overline{C}[R_1, \ldots, R_n]$. $(\overline{R}\theta)\psi\downarrow = (R\theta)\psi\downarrow$ by $(c^i_{\text{fresh}}\theta)\psi\downarrow = (R|_{p_i}\theta)\psi\downarrow$. Furthermore, we have that $\overline{R}\phi_S\downarrow = R'\phi_S\downarrow$ by construction. By $\phi_S \sqsubseteq_s \psi_S$, we get $\overline{R}\psi_S\downarrow = R'\psi_S\downarrow$. By Lemma 4.9, we get $(\overline{R}\theta)\psi\downarrow = (R'\theta)\psi\downarrow$, and thus $(R\theta)\psi\downarrow = (R'\theta)\psi\downarrow$ which is the result we want to prove. □

We can conclude with the proof of our main theorem on equivalence.

THEOREM 3.9. *Let $\mathcal{K}_P$ be a Σ_0^--configuration type-compliant w.r.t. $(\mathcal{T}_0, \delta_0)$ and $\mathcal{K}_Q$ be another Σ_0^--configuration. We have that $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$ w.r.t. Σ_0^- with witness (tr, ϕ) if, and only if, there exists a witness $(\text{tr}', \phi') \in \text{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ of this non-inclusion such that its underlying execution $\mathcal{K}_P \xRightarrow{\text{tr}'} (\mathcal{P}; \phi'; \sigma; i)$ w.r.t. Σ_0^+ is well-typed and $\overline{\text{tr}} = \overline{\text{tr}'}$.*

PROOF. The converse part has already been discussed in Section 3.3.2. Thus, we now prove the direct part. Let $\mathcal{K}_P$ be a Σ_0^--configuration type-compliant w.r.t. $(\mathcal{T}_0, \delta_0)$ and $\mathcal{K}_Q$ be a Σ_0^--configuration. Assume $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$ w.r.t. Σ_0^-. Let $(\text{tr}, \phi) \in \text{trace}_{\Sigma_0^-}(\mathcal{K}_P)$ with underlying substitution σ be a witness

of non-inclusion. Thanks to Proposition 5.4, we know that there exists $(\mathsf{tr}_S, \phi_S) \in \mathsf{trace}_{\Sigma_0^+}(\mathcal{K}_P)$ a witness of this non-inclusion with underlying substitution σ_S such that $\sigma_S = mgu(\Gamma)\rho$, as well as two substitutions λ_P and θ such that:

- $\Gamma = \{(u, v) \mid u, v \in ESt(\mathcal{K}_P)$ such that $u\sigma = v\sigma\}$.
- ρ is a bijective renaming from variables in $dom(\sigma) \setminus dom(mgu(\Gamma))$ to constants in Σ_{fresh} such that $x\rho \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$ if, and only if, $x\sigma$ is an atomic Σ_0^--message.
- $dom(\theta) \subseteq \Sigma_{\mathsf{fresh}}$, for any $c \in \Sigma_{\mathsf{fresh}}$ occurring in tr_S, we have that $c \in dom(\theta)$, $c\theta \in \mathcal{T}(\Sigma, \Sigma_0^- \uplus dom(\phi_S))$, and $\mathsf{w}' \prec c$ for any $\mathsf{w}' \in vars(c\theta)$ (where $\prec$ is the ordering induced by tr_S).
- for any $c \in dom(\theta)$, $(c\theta)\phi_S{\downarrow}$ is a Σ_0^+-message and it is an atom when $c \in \Sigma_{\mathsf{fresh}}^{\mathsf{atom}}$.
- λ_P is the first-order substitution associated to θ through ϕ_S.
- $\phi = \phi_S\lambda_P$, $\sigma = \sigma_S\lambda_P$, and $(\mathsf{tr}_S\theta)\phi{\downarrow} = \mathsf{tr}\phi{\downarrow}$.

First, since $\mathcal{K}_P$ is type-compliant, we know that encrypted subterms in $ESt(\mathcal{K}_P)$ which are unifiable have the same type. Then, by definition of a typing system, this allows us to deduce that $mgu(\Gamma)$ is well-typed. Since we have enough constants of each type, we may assume w.l.o.g. that ρ is also well-typed. Hence, we have that $\sigma_S = mgu(\Gamma)\rho$ is well-typed, which means that (tr_S, ϕ_S) is a well-typed witness of the non-inclusion $\mathcal{K}_P \not\sqsubseteq_t \mathcal{K}_Q$ w.r.t. Σ_0^+. As $(\mathsf{tr}_S\theta)\phi{\downarrow} = \mathsf{tr}\phi{\downarrow}$, we have $\overline{\mathsf{tr}_S} = \overline{\mathsf{tr}}$. This concludes the proof. □

Received