Formal Methods in System Design manuscript No.
(will be inserted by the editor)

Safely composing security protocols

Véronique Cortier · Stéphanie Delaune

Received: date / Accepted: date

Abstract Security protocols are small programs that are executed in hostile environments. Many results and tools have been developed to formally analyze the security of a protocol in the presence of an active attacker that may block, intercept and send new messages. However even when a protocol has been proved secure, there is absolutely no guarantee if the protocol is executed in an environment where other protocols are executed, possibly sharing some common keys like public keys or long-term symmetric keys.

In this paper, we show that security of protocols can be easily composed. More precisely, we show that whenever a protocol is secure, it remains secure even in an environment where arbitrary protocols satisfying a reasonable (syntactic) condition are executed. This result holds for a large class of security properties that encompasses secrecy and various formulations of authentication.

Keywords composition · security protocols · verification

1 Introduction

Security protocols are small programs that aim at securing communications over a public network like the Internet. Considering the increasing size of networks and their dependence on cryptographic protocols, a high level of assurance is needed in the correctness of such protocols. The design of security protocols is difficult and error-prone; many attacks have been discovered even several years after the publication of a protocol. Consequently, there has been a growing interest in applying formal methods

This work has been partly supported by the RNTL project POSÉ and the ARA SSIA Formacrypt.

Véronique Cortier
LORIA, CNRS & INRIA project Cassis, Nancy, France
E-mail: cortier@loria.fr

Stéphanie Delaune
LORIA, CNRS & INRIA project Cassis, Nancy, France
LSV, CNRS & INRIA project Secsi & ENS de Cachan, France
E-mail: delaune@lsv.ens-cachan.fr

for validating cryptographic protocols and many results have been obtained. The main advantage of the formal approach is its relative simplicity which makes it amenable to automated analysis. For example, the secrecy preservation is co-NP-complete for a bounded number of sessions [38], and decidable for an unbounded number of sessions under some additional restrictions (e.g. [27,3,11,18,40]). Many tools have also been developed to automatically verify cryptographic protocols (e.g. [10,7,33,41,39,24]). Of course, considering formal models require the use of abstract models that rules out some practical attacks. Recently, a new line of research is aiming at obtaining higher guarantees using formal approaches (see e.g. [2,8]).

However even when a protocol has been proved secure for an unbounded number of sessions, against a fully active adversary that can intercept, block and send new messages, there is absolutely no guarantee if the protocol is executed in an environment where other protocols are executed, possibly sharing some common keys like public keys or long-term symmetric keys. The interaction with the other protocols may dramatically damage the security of a protocol. Consider for example the two following naive protocols.

$$P_1 : \quad A \to B : \{s\}_{\mathrm{pub}(B)} \qquad\qquad P_2 : \quad \begin{array}{l} A \to B : \{N_a\}_{\mathrm{pub}(B)} \\ B \to A : N_a \end{array}$$

In protocol P_1, the agent A simply sends a secret s encrypted under B's public key. In protocol P_2, the agent sends some fresh nonce to B encrypted under B's public key. The agent B acknowledges A's message by forwarding A's nonce. While P_1 executed alone easily guarantees the secrecy of s, even against active adversaries, the secrecy of s is no more guaranteed when the protocol P_2 is executed. Indeed, an adversary may use the protocol P_2 as an oracle to decrypt any message. More realistic examples illustrating interactions between protocols can be found in e.g. [30].

Main contributions. The purpose of this paper is to investigate sufficient and rather tight conditions for a protocol to be safely used in an environment where other protocols may be executed as well. Our main contribution is to show that whenever a protocol is proved secure when it is executed alone, its security is not compromised by the interactions with any other protocol, provided that any two encrypted sub-messages coming from two different protocol specifications cannot be unified. This can be easily achieved by *tagging* protocols, that is by assigning to each protocol an identifier (e.g. the protocol's name) that should appear in any encrypted message.

We introduce a fragment of the logic PS-LTL (defined in [21]) for which our composition result holds. This fragment allows us to specify a class of security properties that encompasses e.g. secrecy and various formulation of authenticity.

Continuing our example, let us consider the two slightly modified protocols.

$$P_1' : \quad A \to B : \{1, s\}_{\mathrm{pub}(B)} \qquad\qquad P_2' : \quad \begin{array}{l} A \to B : \{2, N_a\}_{\mathrm{pub}(B)} \\ B \to A : N_a \end{array}$$

Our main composition theorem ensures that P_1' can be safely executed together with P_2', without compromising the secrecy of s.

Being able to share keys between protocols is a very desirable property as it allows to save both memory (for storing keys) and time since generating keys is time consuming in particular in the case of public key encryption. We provide in Section 5.4 examples of protocols that share secret materials. For security reasons however, most protocols currently make use of different keys. In this paper, we provide a simple criteria for safely composing protocols that share keys.

The idea of adding an identifier in encrypted messages is not novel. It follows the spirit of the rules proposed in the paper of Abadi and Needham on prudent engineering practice for cryptographic protocols [1] (Principle 10). The use of unique protocol identifiers is also recommended in [30, 14] and has also been used in the design of fail-stop protocols [28]. However, to the best of our knowledge, it has never been proved that it is sufficient for securely executing several protocols in the same environment. Note that some other results also use tags for different purposes. For instance, Blanchet uses tags to exhibit a decidable class [11] but his tagging policy is stronger since any two encrypted subterms in a protocol have to contain different tags. Following our approach, Delaune *et al.* have recently showed [26] that tagging hashes enable to preserve resistance against guessing attacks under composition.

Related work. A result closely related to ours is the one of Guttman and Thayer [29]. They show that two protocols can be safely executed together without damaging interactions, as soon as the protocols are "independent". The independence hypothesis requires in particular that the set of encrypted messages that the two protocols handle should be different. As in our case, this can be ensured by giving each protocol a distinguishing value that should be included in the set of encrypted messages that the protocol handles. However, the major difference with our result is that this hypothesis has to hold not only on the protocol *specification* but also on any valid *execution* of the protocol. In particular, considering again the protocol P_2', an agent should not accept a message of the form $\{2, \{1, m\}_k\}_{\mathrm{pub}(B)}$ while he might not be able to decrypt the inside encryption and detect that it contains the wrong identifier. A more detailed comparison can be found in Section 5.1.

Another result has been recently obtained by Andova *et al.* for a broader class of composition operations [4]. Their result do not allow one to conclude when no typing hypothesis is assumed (that is, when agents are not required to check the type of each component of a message) or for protocols with ciphertext forwarding, that is, when agents have to forward unknown message components.

Datta *et al.* (e.g. [25]) have also studied secure protocol composition in a broader sense: protocols can be composed in parallel, sequentially or protocols may use other protocols as components. However, they do not provide any syntactic conditions for a protocol P to be safely executed in parallel with other protocols. For any protocol P' that might be executed in parallel, they have to prove that the two protocols P and P' satisfy each other invariants. Their approach is thus rather designed for component-based design of protocols.

Our work is also related to those of Canetti *et al.* who study universal composability of protocols [12] in a different context (cryptographic model). They consider composition in a broader sense than our composition result. Indeed, they both enable composition of a protocol with arbitrary other protocols and composition of a protocol with copies of itself. However, in the initial approach [12], they do not allow shared data between protocols, meaning that new encryption and signing keys have to be generated for each component. Composition theorems that allow joint states between protocols are proposed in further work (see e.g. [15, 9, 13]). Several limitations on the applicability of the results are listed in a recent paper of Küsters and Tuengerthal [31]. In [31], the authors propose a new construction for a joint state theorem in the context of asymmetric encryption and signature (no symmetric encryption nor hash as in our formal model). This result allows in particular to compose a protocol with arbitrary protocols using the same kind of construction: they add an identifier in each encryption.

A preliminary version of our results has been presented at FSTTCS'07 [22]. However, in the conference version we prove composability for tagged protocols and secrecy property only. We now consider a weaker hypothesis (non unifiable encrypted messages) and a much larger class of security properties.

Plan of the paper. After some preliminaries (Section 2), we describe the model of protocols in Section 3. In Section 4, we define the logic of security properties for which our composition result holds. Then, in Section 5, we formally state our composition result (Theorem 1) providing examples and discussion. The remaining of the paper is devoted to the proof of this composition result by relying on constraint solving techniques. We first show in Section 6 that we can control the form of minimal attacks. Actually, this result is of independent interest since we provide a decision procedure for solving constraint systems which is more efficient than the one proposed in [23]. Then we explain in Section 7 how to simplify the formula representing the security properties. The final proofs are in Section 8. To ease the understanding of the result, we postpone some of the proofs in the Appendix.

2 Messages and Intruder Capabilities

2.1 Syntax

We use an abstract term algebra to model messages of a protocol. For this we fix several disjoint sets. We consider an infinite set of *agents* $\mathcal{A} = \{a, b, \ldots, \}$, an infinite set of *names* $\mathcal{N} = \{n, m, k, \ldots\}$ and an infinite set of *variables* $\mathcal{X} = \{x, y \ldots, X, Y \ldots\}$. Names are used to model atomic data such as nonces. Cryptographic primitives are represented by *function symbols*. More specifically, we consider the *signature* $\mathcal{F} = \{\mathrm{enc}, \mathrm{enca}, \mathrm{sign}, \langle\,\rangle, \mathsf{init}, \mathrm{h}, \mathrm{priv}\}$ together with arities of the form $\mathrm{ar}(f) = 2$ for the four first symbols and $\mathrm{ar}(f) = 1$ for the two last ones. The symbol init is a special function symbol of arity 0, namely a *constant*. The symbol $\langle\,\rangle$ represents the pairing function. The terms $\mathrm{enc}(m, k)$ and $\mathrm{enca}(m, k)$ represent respectively the message m encrypted with the symmetric (resp. asymmetric) key k whereas the term $\mathrm{sign}(m, k)$ represents the message m signed by the key k. The function symbol h models a hash function and the term $\mathrm{priv}(a)$ represents the private key of the agent a. For simplicity, we confuse the agent names with their public key. The set of Terms is defined inductively by the following grammar:

$$
\begin{array}{rll}
t, t_1, t_2, \ldots ::= & & \text{term} \\
| & x & \text{variable } x \in \mathcal{X} \\
| & a & \text{agent } a \in \mathcal{A} \\
| & n & \text{name } n \in \mathcal{N} \\
| & \mathrm{priv}(a) & \text{application of the symbol priv on } a \in \mathcal{A} \\
| & \mathrm{h}(t) & \text{application of h} \\
| & f(t_1, t_2) & \text{application of symbol } f \in \{\mathrm{enc}, \mathrm{enca}, \mathrm{sign}, \langle\,\rangle\}
\end{array}
$$

We write *vars*(t) (resp. *names*(t), *agents*(t)) for the set of variables (resp. names, agents) occurring in t. A term is *ground* if and only if it has no variable. We write $St(t)$ for the set of *subterms* of a term t. For example, let $t = \mathrm{enc}(\langle a, b\rangle, k)$, we have that

$$
\begin{array}{llll}
\textsf{Pairing} & \textsf{Signature} & \textsf{Hash} & \textsf{Sym./Asym. encryption} \\[4pt]
\dfrac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v\rangle} & \dfrac{T \vdash u \quad T \vdash v}{T \vdash \mathrm{sign}(u, v)} & \dfrac{T \vdash u}{\mathrm{h}(u)} & \dfrac{T \vdash u \quad T \vdash v}{T \vdash f(u, v)}\, f \in \{\mathrm{enc}, \mathrm{enca}\} \\[12pt]
\textsf{1}^{\textsf{st}}\ \textsf{Projection} & \textsf{Verification}\ (\mathit{optional}) & & \textsf{Symmetric decryption} \\[4pt]
\dfrac{T \vdash \langle u, v\rangle}{T \vdash u} & \dfrac{T \vdash \mathrm{sign}(u, v)}{T \vdash u} & & \dfrac{T \vdash \mathrm{enc}(u, v) \quad T \vdash v}{T \vdash u} \\[12pt]
\textsf{2}^{\textsf{nd}}\ \textsf{Projection} & \textsf{Axiom} & & \textsf{Asymmetric decryption} \\[4pt]
\dfrac{T \vdash \langle u, v\rangle}{T \vdash v} & \dfrac{}{T \vdash u}\, u \in T & & \dfrac{T \vdash \mathrm{enca}(u, v) \quad T \vdash \mathrm{priv}(v)}{T \vdash u}
\end{array}
$$

Fig. 1 Intruder deduction system.

$St(t) = \{t, \langle a, b\rangle, a, b, k\}$, $vars(t) = \emptyset$, $names(t) = \{k\}$, $agents(t) = \{a, b\}$. These notions are extended as expected to sets of terms. *Extended names* are names, agents or terms of the form $\mathrm{priv}(a)$. The set of *Extended names* associated to a term t, denoted $\mathrm{n}(t)$, is

$$\mathrm{n}(t) = names(t) \cup \{a, \mathrm{priv}(a) \mid a \in agents(t)\}.$$

For example, we have that $\mathrm{n}(\mathrm{enc}(n, a)) = \{n, a, \mathrm{priv}(a)\}$. Substitutions are written $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ with $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$. The substitution σ is *closed* if and only if all the t_i are ground. The application of a substitution σ to a term t is written $\sigma(t)$ or $t\sigma$. Two terms t_1 and t_2 are *unifiable* if $t_1\sigma = t_2\sigma$ for some substitution σ, otherwise there are *non-unifiable*. Lastly, we assume a set $\mathcal{P}$ of predicates together with their arities.

2.2 Intruder capabilities

The ability of the intruder is modeled by a deduction system described in Figure 1 and corresponds to the usual Dolev-Yao rules. The first line describes the *composition* rules. The two last lines describe the *decomposition* rules and the axiom. Intuitively, these deduction rules say that an intruder can compose messages by pairing, signing, hashing, encrypting messages provided he has the corresponding keys. Conversely, it can decompose messages by projecting or decrypting provided it has the decryption keys. For signatures, the intruder is also able to *verify* whether a signature $\mathrm{sign}(m, k)$ and a message m match (provided she has the verification key), but this does not give her any new message. That is why this capability is not represented in the deduction system. We also consider an optional rule (Verification)

$$\frac{T \vdash \mathrm{sign}(u, v)}{T \vdash u}$$

that expresses that an intruder can retrieve the whole message from its signature. This property may or may not hold depending on the signature scheme, and that is why this rule is optional. Our results hold in both cases (that is, when the deduction relation $\vdash$ is defined with or without this rule).

A term u is *deducible* from a set of terms T, denoted by $T \vdash u$ if there exists a *proof*, i.e. a tree such that the root is $T \vdash u$, the leaves are of the form $T \vdash v$ with $v \in T$ (*axiom* rule) and every intermediate node is an instance of one of the rules of the deduction system.

Example 1 The term $\langle k_1, k_2 \rangle$ is deducible from the set $T_1 = \{\text{enc}(k_1, k_2), k_2\}$. A proof of $T_1 \vdash \langle k_1, k_2 \rangle$ is:

$$\dfrac{\dfrac{T_1 \vdash \text{enc}(k_1, k_2) \quad T_1 \vdash k_2}{T_1 \vdash k_1} \qquad T_1 \vdash k_2}{T_1 \vdash \langle k_1, k_2 \rangle}$$

3 Models for security protocols

In this section we give a language for specifying protocols and define their execution in the presence of an active adversary.

3.1 Syntax

We consider protocols specified in a language allowing parties to exchange messages built from identities and randomly generated nonces using public key, symmetric encryption and digital signatures. The individual behavior of each protocol participant is defined by a *role* describing a sequence of *events*. The main events we consider are *communication events* (i.e. message receptions and message transmissions) and *new events* to model random numbers generation. To be able to specify a large class of security properties (a logic of properties is given in Section 4), we also consider *status events*. Those events are issued by participant to denote their current state in the execution of a protocol role.

Definition 1 (event) An *event* is one of the following:

- a *communication event*, i.e. a message reception, denoted by $\mathsf{rcv}(m)$ or a message transmission, denoted by $\mathsf{snd}(m)$, where m is a term; or
- a *new event*, denoted by $\mathsf{new}\ X$ where X is a variable; or
- a *status event* of the form $P(t_1, \ldots, t_n)$ where each t_i is a term (not necessarily ground) and $P \in \mathcal{P}$ is a predicate symbol of arity n.

Typically status events give information about the state of the principal. For instance, we will consider a status event that indicates that the principal has started or finished an execution. The set of variables of an event is defined as expected, considering all the terms occurring in the event's specification.

Definition 2 (roles) A role is a finite sequence of events $\mathsf{e}_1; \ldots; \mathsf{e}_\ell$ such that

1. for any sent or status event e_i, for any variable $x \in \mathit{vars}(\mathsf{e}_i)$, we have that $x \in \bigcup_{1 \leq j < i} \mathit{vars}(\mathsf{e}_j)$, and
2. a variable occurring in a new event does not appear previously in the sequence.

The *length* of a role is the number of events in its sequence.

We denote by Roles the set of roles. A k-party protocol is given by k such a role. More formally, a *k-party protocol* is a mapping $\Pi : [k] \to \mathsf{Roles}$, where $[k] = \{1, 2, \ldots, k\}$. The condition stated in Definition 2 ensures that each variable which appears in a sent or status event is either a nonce or has been introduced in a previously received message. The set of variables, names or extended names of a protocol is defined as expected, considering all the terms occurring in the role's specification.

The *composition* of two protocols Π_1 and Π_2, denoted by $\Pi_1 \mid \Pi_2$ is simply the protocol obtained by the union of the roles of Π_1 and Π_2. If $\Pi_1 : [k_1] \to \mathsf{Roles}$ and $\Pi_2 : [k_2] \to \mathsf{Roles}$, then $\Pi = \Pi_1 \mid \Pi_2 : [k_1 + k_2] \to \mathsf{Roles}$ with $\Pi(i) = \Pi_1(i)$ for any $1 \leq i \leq k_1$ and $\Pi(k_1 + i) = \Pi_2(i)$ for any $1 \leq i \leq k_2$.

Example 2 Consider the famous Needham-Schroeder protocol [37].

$$
\begin{array}{rl}
A \to B : & \{N_a, A\}_{\mathrm{pub}(B)} \\
B \to A : & \{N_a, N_b\}_{\mathrm{pub}(A)} \\
A \to B : & \{N_b\}_{\mathrm{pub}(B)}
\end{array}
$$

The agent A sends to B his name and a fresh nonce (a randomly generated value) encrypted with the public key of B. The agent B answers by copying A's nonce and adds a fresh nonce N_B, encrypted by A's public key. The agent A acknowledges by forwarding B's nonce encrypted by B's public key. For instance, let a, b, and c be three agents. The role $\Pi(1)$ corresponding to the first participant played by a talking to c and the role $\Pi(2)$ corresponding to the second participant played by b with a are described below.

$$
\begin{array}{rlcrl}
\Pi(1) := & \mathsf{new}\ X; & \qquad & \Pi(2) := & \mathsf{rcv}(\mathrm{enca}(\langle y, a\rangle, b)); \\
& \mathsf{snd}(\mathrm{enca}(\langle X, a\rangle, c)); & & & \mathsf{new}\ Y; \\
& \mathsf{rcv}(\mathrm{enca}(\langle X, x\rangle, a)); & & & \mathsf{snd}(\mathrm{enca}(\langle y, Y\rangle, a)); \\
& \mathsf{snd}(\mathrm{enca}(x, c)). & & & \mathsf{rcv}(\mathrm{enca}(Y, b)).
\end{array}
$$

Note that, since our definition of role is not parametric, we have also to consider a role corresponding to the first participant played by a talking to b for example. If more agent identities need to be considered, then the corresponding roles should be added to the protocol. It has been shown however that two agents are sufficient (one honest and one dishonest) for proving security properties such as those we consider in this paper [19]. In this example, we chose to not use status event. Actually, they are meaningful to specify the security properties and have no real interest for the description of the protocol itself. We will illustrate the usefulness of status events in Section 4.

Clearly, not all protocols written using the syntax above are meaningful. In particular, some of them might not be *executable*. For instance, a k-party protocol where $\Pi(1) := \mathsf{rcv}(\mathrm{h}(x)); \mathsf{snd}(x)$ is not executable since an agent is not able to extract the content of a hash. A precise definition of executability is not relevant for our result. We use instead a weaker hypothesis (see Theorem 1, Condition 2). In particular, our combination result also holds for non executable protocols such as the one given above.

3.2 Semantics

We start with the description of the execution model of the protocol in the presence of an active attacker. The model we consider is rather standard. The parties in the system execute a (potentially unbounded) number of protocol sessions with each other. A role may be executed in several sessions, using different nonces at each session. Moreover, since the adversary may block, redirect and send new messages, all the sessions might be interleaved in many ways. This is captured by the notion of *scenario*.

Definition 3 (scenario) A *scenario for a protocol* $\Pi : [k] \to \mathsf{Roles}$ is a sequence $\mathsf{sc} = (r_1, s_1) \cdots (r_n, s_n)$ such that $1 \leq r_i \leq k$, $s_i \in \mathbb{N}$, the number of identical occurrences of a pair (r, s) is smaller than the length of the role r, and whenever $s_i = s_j$ then $r_i = r_j$.

The numbers r_i and s_i represent respectively the involved role and the session number. An occurrence of (r, s) in sc means that the role r of session s executes its next action. The condition on the number of occurrences of a pair ensures that such an action is always available. The last condition ensures that a session number is not reused on other roles.

Let $\Pi = \Pi_1 \mid \Pi_2$ be a protocol obtained by composition of Π_1 and Π_2 and let sc be a scenario for Π. The scenario $\mathsf{sc}|_{\Pi_1}$ is simply the sequence obtained from sc by removing any element (r, s) where r is a role of Π_2.

Given a protocol Π and a scenario sc, we can define the symbolic trace, i.e. a sequence of events, associated to Π and sc. It corresponds to the sequence of events in the order defined by the scenario. Variables occurring in new events are instantiated by fresh names while the other variables are left unchanged. This symbolic trace represents a potentially infinite number of concrete traces. Intuitively, the variables can be instantiated in potentially infinite ways, depending on the messages sent by the intruder. A trace is said to be *ground* if it contains no variable.

Definition 4 (symbolic trace associated to Π and sc) Given a scenario

$$\mathsf{sc} = (r_1, s_1) \cdots (r_n, s_n)$$

for a k-party protocol Π, the *symbolic trace* $\mathsf{tr} = \mathsf{e}_1; \ldots; \mathsf{e}_\ell$ *associated to* sc is defined as follows. Let $\Pi(j) = \mathsf{e}_1^j; \ldots; \mathsf{e}_{k_j}^j$ for $1 \leq j \leq k$. Let $p_i = \#\{(r_j, s_j) \in \mathsf{sc} \mid j \leq i, s_j = s_i\}$, i.e. the number of previous occurrences in sc of the session s_i. We have $p_i \leq k_{r_i}$ and $\mathsf{e}_i = \mathsf{e}_{p_i}^{r_i} \sigma_{r_i, s_i}$ where

- $\mathrm{dom}(\sigma_{r,s}) = \{vars(\mathsf{e}_i^r) \mid 1 \leq i \leq k_r$ and e_i^r is a new or a received event$\}$, i.e. variables occurring in $\Pi(r)$,
- $\sigma_{r,s}(X) = n_{X,s}$ if $X \in \{Y \mid 1 \leq i \leq k_r$ and $\mathsf{e}_i^r = \mathsf{new}\ Y\}$, where $n_{X,s} \in \mathcal{N}$.
- $\sigma_{r,s}(x) = x_s$ otherwise, where x_s is a variable.

We assume that the names $n_{x,s}$ and the variables x_s are fresh, that is, they are supposed not to occur in any other protocol or security formula.

Example 3 Consider again the Needham-Schroeder protocol. Let $\Pi(1)$ and $\Pi(2)$ be the two roles introduced in Example 2. Let s_1 and s_2 be two sessions numbers ($s_1 \neq s_2$) and $\mathsf{sc} = (1, s_1)(1, s_1)(2, s_2)(2, s_2)(2, s_2)(1, s_1)(1, s_1)$. This is the scenario allowing us to retrieve the famous attack due to Lowe [32]. The symbolic trace associated to Π and sc is given below:

$$\begin{aligned}\mathsf{tr} = {} & \mathsf{new}\ n_{X,s_1};\ \mathsf{snd}(\mathrm{enca}(\langle n_{X,s_1}, a\rangle, c));\\ & \mathsf{rcv}(\mathrm{enca}(\langle y_{s_2}, a\rangle, b));\ \mathsf{new}\ n_{Y,s_2};\ \mathsf{snd}(\mathrm{enca}(\langle y_{s_2}, n_{Y,s_2}\rangle, a));\\ & \mathsf{rcv}(\mathrm{enca}(\langle n_{X,s_1}, x_{s_1}\rangle, a));\ \mathsf{snd}(\mathrm{enca}(x_{s_1}, c))\end{aligned}$$

Appending an event e to a trace tr is written $\mathsf{tr};\ \mathsf{e}$. The function length has the usual meaning: $\mathsf{length}([\,]) = 0$ and $\mathsf{length}(\mathsf{tr};\ \mathsf{e}) = 1 + \mathsf{length}(\mathsf{tr})$. The prefix trace consisting of the first i events is denoted as tr_i, with $\mathsf{tr}_0 = [\,]$ and $\mathsf{tr}_n = \mathsf{tr}$ when $n \geq \mathsf{length}(\mathsf{tr})$.

Definition 5 (knowledge of a trace tr) Let tr be a trace. The knowledge of tr is the set of terms given by $\mathsf{K}(\mathsf{tr}) = \{\mathsf{init}\} \cup \{u \mid \mathsf{snd}(u) \in \mathsf{tr}\}$.

An *execution trace* is an instance of such a symbolic trace. As usual, we are only interested in *valid* execution traces - those traces where the attacker only sends messages that he can compute from his knowledge and the messages he has seen on the network.

Definition 6 (valid execution trace) Let T_0 be a finite set of ground terms (intuitively T_0 represents the initial knowledge of the attacker). A ground execution trace $\mathsf{tr} = \mathsf{e}_1; \ldots; \mathsf{e}_\ell$ is *valid* w.r.t. T_0 if for all $1 \leq i \leq \ell$, whenever $\mathsf{e}_i = \mathsf{rcv}(m)$, we have that $T_0 \cup \mathsf{K}(\mathsf{tr}_i) \vdash m$.

Example 4 Let $T_0 = \{a, b, c, \mathrm{priv}(c)\}$. Let tr be the symbolic trace described in Example 3 and $\sigma = \{y_{s_2} \mapsto n_{X,s_1},\ x_{s_1} \mapsto n_{Y,s_2}\}$. The trace $\mathsf{tr}\sigma$ is valid w.r.t. T_0. Indeed, we have that

- $T_1 \stackrel{\mathsf{def}}{=} T_0, \mathsf{init}, \mathrm{enca}(\langle n_{X,s_1}, a\rangle, c) \vdash \mathrm{enca}(\langle n_{X,s_1}, a\rangle, b)$, and
- $T_1, \mathrm{enca}(\langle n_{X,s_1}, n_{Y,s_2}\rangle, a) \vdash \mathrm{enca}(\langle n_{X,s_1}, n_{Y,s_2}\rangle, a)$.

In the next section, we define what it means for a protocol to satisfy a security property. We introduce a logic for properties that encompasses classical properties like secrecy and authentication.

4 Security Properties

In this section, we review a logic, called PS-LTL, for specifying security properties. This logic is actually a (syntactic) fragment of the logic proposed in [21]. The logic is based on linear temporal logic (LTL) with pure-past operators. PS-LTL provides adequate flexibility, allowing one to specify several security properties like secrecy and different forms of authentication among them aliveness, weak agreement and non-injective agreement. Its semantics is defined as usual on execution traces.

4.1 PS-LTL: Syntax and Semantics

Compared to [21], we split off the status events (defined with predicates) from the communication events (send, received or new events). Indeed, the first kind of events are used to specify security properties while the others are internal events describing the execution of the protocol. The temporal operators should only concern status events.

That is why we divided the logic into two layers and (slightly) change the semantics accordingly. The first layer consists in formula made up from status event, temporal operators and the classical $\neg, \vee, \wedge, \exists$, and $\forall$ logical operators. The second layer consists in formula made up from the first layer, the special predicate learn and the classical $\neg, \vee, \wedge, \exists$, and $\forall$ logical operators.

Definition 7 (PS-LTL formula) A PS-LTL formula ϕ, is defined by the following grammar:

$$\psi, \psi_i := \mathsf{true} \mid P(t_1, \ldots, t_n) \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \mathrm{Y}\psi \mid \psi_1 \,\mathrm{S}\, \psi_2 \mid \exists x.\psi \mid \forall x.\psi$$

$$\phi, \phi_i := \psi \mid \mathsf{learn}(m) \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \exists x.\phi \mid \forall x.\phi$$

where the t_i's and m are terms (not necessarily ground)

Standard formulas true, $\neg\phi$, $\phi \wedge \phi$ and $\phi \vee \phi$ carry the usual meaning. The formula $\mathsf{learn}(m)$ states that the attacker knows the term m whereas $P(t_1, \ldots, t_n)$ is a status event. The formula $\mathrm{Y}\psi$ means that 'yesterday ψ held', while $\psi_1 \,\mathrm{S}\, \psi_2$ means that 'ψ_1 held *ever since* a moment in which ψ_2 held'. When x is a variable, we write $\exists x.\phi$ and $\forall x.\phi$ to bind x in ϕ, with the quantifiers carrying the usual meaning. Other operators can be represented using the above defined operators. The abbreviations false and $\Rightarrow$ are defined by $\mathsf{false} \stackrel{\mathsf{def}}{=} \neg\mathsf{true}$ and $\phi_1 \Rightarrow \phi_2 \stackrel{\mathsf{def}}{=} \neg\phi_1 \vee \phi_2$. We also used $\Diamond\, \psi$ as a shorthand for $\mathsf{true}\, \mathrm{S}\, \psi$.

In the sequel, we assume that PS-LTL formulas are *closed*, i.e. they contain no free variables, and that each variable is quantified at most once (this can be easily ensured by using renaming). We also assume that the variables occurring in a formula ϕ are disjoint from the variables occurring in the considered execution trace tr. Given a trace tr, we denote by $\overline{\mathsf{tr}}$ the sequence of status events obtained by removing from tr all the communication and new events. PS-LTL formulas are interpreted at some position along a trace as stated in Definition 8.

Definition 8 (concrete validity) Let ϕ be a closed PS-LTL formula, tr be a ground execution trace and T_0 be a finite set of ground terms. We define $\langle \mathsf{tr}, T_0 \rangle \models \phi$ as:

$\langle \mathsf{tr}, T_0 \rangle \models \mathsf{true}$		
$\langle \mathsf{tr}, T_0 \rangle \models \mathsf{learn}(m)$	iff	$T_0 \cup \mathsf{K}(\mathsf{tr}) \cup \vdash m$
$\langle \mathsf{tr}, T_0 \rangle \models \neg\phi$	iff	$\langle \mathsf{tr}, T_0 \rangle \not\models \phi$
$\langle \mathsf{tr}, T_0 \rangle \models \phi_1 \wedge \phi_2$	iff	$\langle \mathsf{tr}, T_0 \rangle \models \phi_1$ and $\langle \mathsf{tr}, T_0 \rangle \models \phi_2$
$\langle \mathsf{tr}, T_0 \rangle \models \phi_1 \vee \phi_2$	iff	$\langle \mathsf{tr}, T_0 \rangle \models \phi_1$ or $\langle \mathsf{tr}, T_0 \rangle \models \phi_2$
$\langle \mathsf{tr}, T_0 \rangle \models \exists x.\phi$	iff	there exists a ground term t such that $\langle \mathsf{tr}, T_0 \rangle \models \phi[x \mapsto t]$
$\langle \mathsf{tr}, T_0 \rangle \models \forall x.\phi$	iff	for all ground term t we have that $\langle \mathsf{tr}, T_0 \rangle \models \phi[x \mapsto t]$

For the temporal formulas, only the status events are meaningful.

$\langle \mathsf{tr}, T_0 \rangle \models P(t_1, \ldots, t_n)$	iff	$\overline{\mathsf{tr}} = \mathsf{tr}'; P(t_1, \ldots, t_n)$
$\langle \mathsf{tr}, T_0 \rangle \models \mathrm{Y}\psi$	iff	$\overline{\mathsf{tr}} = \mathsf{tr}'; \mathsf{e}$ and $\langle \mathsf{tr}', T_0 \rangle \models \psi$
$\langle \mathsf{tr}, T_0 \rangle \models \psi_1 \,\mathrm{S}\, \psi_2$	iff	$\exists i \in [0, \mathsf{length}(\overline{\mathsf{tr}})]$ such that – $\langle \overline{\mathsf{tr}}_i, T_0 \rangle \models \psi_2, and$ – $\forall j \in [i+1, \mathsf{length}(\overline{\mathsf{tr}})]$, we have $\langle \overline{\mathsf{tr}}_j, T_0 \rangle \models \psi_1$

We now define the subset of PS-LTL formulas, namely PS-LTL^-, over which our composition result holds. We say a PS-LTL formula is *quantifier-free* if it does not contain any quantifier (neither $\exists$, nor $\forall$). We will only consider security formulas of the form $\forall x_1. \dots .\forall x_n.\phi'$ where ϕ' is quantifier-free. Note that, this means that the attack formulas we consider are those of the form $\exists x_1, \dots, \exists x_n.\neg\phi'$. We also need to control the occurrences of $\mathsf{learn}(m)$. We say that a formula ϕ is *positive* (resp. *negative*) if every occurrence of $\mathsf{learn}(m)$ in ϕ appears under an even (resp. odd) number of negations. This restriction allows us to avoid negated deducibility constraints.

Definition 9 (PS-LTL^+, PS-LTL^-) We say that ϕ is a *universal negative formula* (resp. *existential positive formula*) if ϕ is of the form $\forall x_1. \dots .\forall x_n.\phi'$ (resp. $\exists x_1. \dots .\exists x_n.\phi'$) where ϕ' is quantifier-free and negative (resp. positive). We denote by PS-LTL^- (resp. PS-LTL^+) such a fragment.

In the remainder, we consider universal negative security formulas (PS-LTL^- fragment), i.e. existential positive attack formulas (PS-LTL^+ fragment).

Definition 10 ($\Pi \models \phi$) Let ϕ be a closed PS-LTL formula, tr be a symbolic trace and T_0 be a set of ground terms. We say that $\langle \mathsf{tr}, T_0\rangle \models \phi$ if $\langle \mathsf{tr}\sigma, T_0\rangle \models \phi$ for every substitution σ such that $\mathsf{tr}\sigma$ is a valid execution trace.

Let Π be a protocol and T_0 be a set of ground terms. We say that $\Pi \models \phi$ w.r.t. T_0, if $\langle \mathsf{tr}, T_0\rangle \models \phi$ for all symbolic trace tr associated to some scenario of Π.

4.2 Writing Security Properties with PS-LTL

In this section, we show how to specify several security properties in PS-LTL^-. We illustrate this with the Needham-Schroeder protocol presented in Example 2.

Secrecy. We can easily specify the standard notion of secrecy, which is the inability of an attacker to obtain the value of the secret. The secrecy of a long-term key, e.g. $\mathrm{priv}(a)$, can be checked by the PS-LTL^- formula $\neg\mathsf{learn}(\mathrm{priv}(a))$. We can also express the secrecy of a nonce, e.g. the nonce generated by b for a in the role $\Pi(2)$ described in Example 2. For this, we have to introduce a status event, say nonce. Thus, we modify the role of $\Pi(2)$ by adding the status event $\mathsf{nonce}(Y)$ just after the event $\mathsf{new}\ Y$. We obtain the two following roles:

$$\begin{array}{rl} \Pi(1) := & \mathsf{new}\ X; \\ & \mathsf{snd}(\mathrm{enca}(\langle X, a\rangle, c)); \\ & \mathsf{rcv}(\mathrm{enca}(\langle X, x\rangle, a)); \\ & \mathsf{snd}(\mathrm{enca}(x, c)). \end{array} \qquad \begin{array}{rl} \Pi(2) := & \mathsf{rcv}(\mathrm{enca}(\langle y, a\rangle, b)); \\ & \mathsf{new}\ Y; \\ & \mathsf{nonce}(Y); \\ & \mathsf{snd}(\mathrm{enca}(\langle y, Y\rangle, a)); \\ & \mathsf{rcv}(\mathrm{enca}(Y, b)). \end{array}$$

Thus, now, we can require that the nonces generated by b for a has to be kept secret. This can be done by the following PS-LTL^- formula

$$\forall x. (\Diamond\, \mathsf{nonce}(x)) \Rightarrow \neg\mathsf{learn}(x).$$

Consider the trace tr' obtained from tr (described in Example 3) by inserting the status event $\mathsf{nonce}(n_{Y,s_2})$ just after the event $\mathsf{new}\ n_{Y,s_2}$, i.e.

$$\begin{array}{rl} \mathsf{tr}' = & \mathsf{new}\ n_{X,s_1}; \mathsf{snd}(\mathrm{enca}(\langle n_{X,s_1}, a\rangle, c)); \mathsf{rcv}(\mathrm{enca}(\langle y_{s_2}, a\rangle, b)); \\ & \mathsf{new}\ n_{Y,s_2}; \mathsf{nonce}(n_{Y,s_2}); \mathsf{snd}(\mathrm{enca}(\langle y_{s_2}, n_{Y,s_2}\rangle, a)); \\ & \mathsf{rcv}(\mathrm{enca}(\langle n_{X,s_1}, x_{s_1}\rangle, a)); \mathsf{snd}(\mathrm{enca}(x_{s_1}, c)) \end{array}$$

Consider the substitution σ and the set of ground terms T_0 given in Example 4. We have that $\langle \mathsf{tr}\sigma, T_0\rangle \models \exists x.(\Diamond\, \mathsf{nonce}(x)) \wedge \mathsf{learn}(x)$. It is indeed easy to see that $\langle \mathsf{tr}\sigma, T_0\rangle \models (\Diamond\, \mathsf{nonce}(n_{Y,s_2}))$, and $\langle \mathsf{tr}\sigma, T_0\rangle \models \mathsf{learn}(n_{Y,s_2})$. This means that the protocol Π (modified version) does not satisfies the secrecy property stated above.

Temporary secret. Some protocols have to preserve secrecy, but only during a period. For instance, in e-voting, it should be impossible for an attacker to learn a vote before the opening phase. This can be easily expressed in PS-LTL^{-}. For this, we insert a status event $\mathsf{opening}$ in the protocol specification just before the beginning of the opening phase. Then the formula can be written as follows:

$$\forall x.\, (\Diamond\, \mathsf{vote}(x) \wedge \neg\Diamond\, \mathsf{opening}) \Rightarrow \neg\mathsf{learn}(x).$$

Fairness. In a fair contract signing protocols two agents A and B want to exchange their signatures on a contract in such a way that at the end of the protocol either both participants obtain the signed contract or none of them does so. This kind of protocols either terminates in a final state where the exchange has been aborted or in a final state where the exchange did succeed. For the purpose of our example, we suppose that the role modelling the participant P (either A or B) is annotated as follows: the event $\mathsf{Pend}(c)$ indicates that the role is in a final state for some contract c; the event $\mathsf{Pcontract}(c)$ indicates that P successfully received the signed contract. Then, fairness for A can be modelled as

$$\forall x.\, \mathsf{Aend}(x) \Rightarrow (\Diamond \mathsf{Acontract}(x) \vee \neg\Diamond \mathsf{Bcontract}(x))$$

The formula says that for any contract whenever A is in a final state ($\mathsf{Aend}(x)$), either A did obtain the contract signed by B (formalized by the formula $\Diamond \mathsf{Acontract}(x)$), or B did not obtain the contract signed by A (formalized by the formula $\neg\Diamond \mathsf{Bcontract}(x)$).

We also cover various form of authentication except injective agreement, which would require counting events in a trace. This would require an extension of the logic.

Aliveness. This property is the weakest form of authentication in Lowe's hierarchy [34].

> A protocol satisfies *aliveness* if, whenever an honest agent completes a run of the protocol, apparently with another honest agent B, then B has previously run the protocol.

Note that B may not necessarily believe that he was running the protocol with A. Also, B may not have run the protocol *recently*. The aliveness of principal B to initiator A can be specified in our formalism. First, we have to consider two status events start and end. We insert them at the beginning and at the end of each role respectively. For instance, in $\Pi(1)$, we insert $\mathsf{start}(a)$ at the beginning and $\mathsf{end}(a, c)$ at the end. This expresses the fact that the role is executed by a with c. We insert $\mathsf{start}(b)$ and $\mathsf{end}(b, a)$ in $\Pi(2)$. Now, the aliveness property can be specified as follows:

$$(\mathsf{end}(a, b) \Rightarrow \Diamond\, \mathsf{start}(b)) \wedge (\mathsf{end}(b, a) \Rightarrow \Diamond\, \mathsf{start}(a))$$

This corresponds to the fact that the property $\mathsf{end}(x, y) \Rightarrow \mathsf{start}(y)$ has to be satisfied when x and y are both honest agents. For the Needham-Schroeder public-key protocol, the aliveness property is satisfied.

Weak agreement. Weak agreement is slightly stronger than aliveness.

> A protocol guarantees *weak agreement* if, whenever an honest agent completes a run of the protocol, apparently with another honest agent B, then B has previously been running the protocol, apparently with A.

The weak agreement property can also been expressed in our formalism. We have again to add status events start and end in our specification. However, the predicate start will have also two parameters: $\mathsf{start}(a, c)$ expresses the fact that a has started a session with c. Now, the weak agreement property can be specified as follows:

$$(\mathsf{end}(a, b) \Rightarrow \Diamond\, \mathsf{start}(b, a)) \wedge (\mathsf{end}(b, a) \Rightarrow \Diamond\, \mathsf{start}(a, b))$$

For the Needham-Schroeder public-key protocol, it is well-known that this property is not satisfied: b can complete a session apparently with a whereas a has never started a session with b.

We can also express some refinements of these properties by distinguishing the case where an agent starts a session as an initiator or as a responder. Moreover, we can also express the fact that the two agents agreed on some data D. This allows us to express the non-injective agreement security property.

5 Composition result

Even if a protocol is secure for an unbounded number of sessions, its security may collapse if the protocol is executed in an environment where other protocols sharing some common keys are executed. A first example has been informally given in Introduction. In Sections 5.1 and 5.2, we introduce and discuss the hypotheses we need to safely compose protocols, providing counter-examples that justify the necessity of our hypotheses. Our main result is formally stated in Section 5.3.

5.1 Disjoint encryption

To avoid a ciphertext from a protocol Π_1 to be decrypted in another protocol Π_2, we consider protocols that satisfies *disjoint encryption*. This notion is formally defined below (see Definition 11) and relies on the following notion of *encrypted subterms*.

An *encrypted term* is a term of the form $\mathrm{enc}(u, v)$, $\mathrm{enca}(u, v)$, $\mathrm{sign}(u, v)$ for some terms u, v or $\mathrm{h}(u)$ for some term u. Given a set of terms T, we denote by $\mathit{EncSt}(T)$ the set of encrypted subterms of T, i.e.

$$\mathit{EncSt}(T) = \{t' \in \mathit{St}(T) \mid t \text{ is an encrypted term}\}.$$

This notation is extended as expected to events and PS-LTL formula. Given a protocol Π, consider the substitution σ such that $\mathrm{dom}(\sigma) = \{X \mid \mathsf{new}\, X \in \Pi\}$ and $X\sigma = n_X$ for any $X \in \mathrm{dom}(\sigma)$. We define $\mathit{EncSt}(\Pi)$ as follows:

$$\mathit{EncSt}(\Pi) = \{\mathit{EncSt}(\mathsf{e})\sigma \mid \mathsf{e} \in \Pi\}.$$

Note that we instantiate the variables under new events. This reflects that parties will check the nonces they have generated on their own. For example, consider the two following protocols. Let Π_1 be a protocol with only one role:

$$\Pi_1(1) := \mathsf{new}\ X;\ \mathsf{snd}(\mathrm{enca}(X, a));\ \mathsf{rcv}(\mathrm{enca}(\langle X, X\rangle, a));\ \mathsf{Fail}$$

The agent sends to itself a message of the form $\mathrm{enca}(N, \mathrm{pub}(A))$ and waits for $\mathrm{enca}(\langle N, N\rangle, \mathrm{pub}(A))$, in which case he raises the status event Fail, where Fail is a predicate of arity 0. The protocol Π_1 will never reach the status event Fail. Let now Π_2 be a protocol with only one role:

$$\Pi_2(1) := \mathsf{new}\ Y;\ \mathsf{snd}(\mathrm{enca}(\langle Y, Y\rangle, a))$$

Even if Π_1 is composed with Π_2, Π_1 will never reach the status event Fail. However, if we did not instantiate variables under new events, the two encrypted terms $\mathrm{enca}(\langle X, X\rangle, a)$ and $\mathrm{enca}(\langle Y, Y\rangle, a)$ would be unifiable.

Definition 11 (disjoint encryption) Let T_1 and T_2 be two sets of terms. We say that T_1 and T_2 have *disjoint encryption* if $\mathit{vars}(T_1) \cap \mathit{vars}(T_2) = \emptyset$ and for every encrypted terms $t_1' \in \mathit{EncSt}(T_1)$ and $t_2' \in \mathit{EncSt}(T_2)$, we have that t_1' and t_2' are non-unifiable.

Two protocols Π_1 and Π_2 (we assume that they do not share any variable) have *disjoint encryption* if $\mathit{EncSt}(\Pi_1)$ and $\mathit{EncSt}(\Pi_2)$ have disjoint encryption.

Example 5 The role $\Pi(1)$ and $\Pi(2)$ described in Example 2 do not have disjoint encryption since $\mathrm{enca}(\langle n_X, x\rangle, a)$ and $\mathrm{enca}(\langle y, n_Y\rangle, a)$ are unifiable. Anyway, we know that these two roles can not be safely composed (Lowe's attack). However, two protocols having disjoint key material, e.g. Needham-Schroeder-Lowe public key protocol and Needham-Schroeder symmetric key protocol have disjoint encryption.

However, protocols that use common keys (e.g. common public keys) may not enjoy the disjoint encryption property. A way to force disjoint encryption is to use tag. Requiring that two protocols satisfy disjoint encryption can be very easily achieved in practice: it is sufficient for example to add the name of the protocol in each encrypted term.

Definition 12 (well-tag, α-tag) Let α be a ground term. A term t is α-tagged if $\mathit{EncSt}(t) \subseteq \{f(\langle \alpha, t_1\rangle, t_2),\ \mathrm{h}(\langle \alpha, t_1\rangle) \mid f \in \{\mathrm{enc}, \mathrm{enca}, \mathrm{sign}\}, t_1, t_2 \in \mathsf{Terms}\}$. A term is said to be *well-tagged* if it is α-tagged for some ground term α.

A protocol Π is α-tagged is any term occurring in the role of the protocol is α-tagged. A protocol is said *well-tagged* if it is α-tagged for some ground term α.

The following proposition is an easy consequence of the previous definition since two terms which are respectively α and β-tagged ($\alpha \neq \beta$) have necessarily disjoint encryption.

Proposition 1 *Let Π_1 and Π_2 be two well-tagged protocols such that Π_1 is α-tagged and Π_2 is β-tagged with $\alpha \neq \beta$. Then the protocols Π_1 and Π_2 have disjoint encryption.*

Proof Since Π_1 and Π_2 are respectively α-tagged and β-tagged, we have that

- $\mathit{EncSt}(\Pi_1) \subseteq \{f(\langle \alpha, t_1\rangle, t_2),\ \mathrm{h}(\langle \alpha, t_1\rangle) \mid f \in \{\mathrm{enc}, \mathrm{enca}, \mathrm{sign}\}, t_1, t_2 \in \mathsf{Terms}\}$,

– $EncSt(\Pi_2) \subseteq \{f(\langle\beta, t_1\rangle, t_2),\ \mathrm{h}(\langle\beta, t_1\rangle) \mid f \in \{\mathrm{enc}, \mathrm{enca}, \mathrm{sign}\}, t_1, t_2 \in \mathsf{Terms}\}$.

Now, since α and β are not unifiable, it is easy to conclude. □

Note that (as opposite to [29]) we do not require that the agents check that nested encrypted terms are correctly tagged. For example, let Π be a protocol with one role as follows:

$$\Pi(1) = \mathsf{rcv}(\mathrm{enca}(\langle\alpha, x\rangle, a));\ \mathsf{snd}(\mathrm{enca}(\langle\alpha, x\rangle, b)).$$

The message $\mathrm{enca}(\langle\alpha, \mathrm{enc}(a, k)\rangle, a)$ (which is not correctly tagged) would be accepted by the agent playing the role.

5.2 Controlling the position of critical long-term keys

Disjoint encryption is not a sufficient condition. Indeed critical long-term keys should not be revealed in clear. Consider for example the following two protocols. Note that they satisfy disjoint encryption since P_4 has no encrypted subterm.

$$P_3: \quad A \to B : \{\alpha, s\}_{k_{ab}} \qquad\qquad P_4: \quad A \to B : k_{ab}$$

The security of protocol P_3 is compromised by the execution of P_4. Thus we will require that long-term keys (except possibly the public ones) do not occur in plaintext in the protocol. This is not a real restriction since not disclosing the long term private keys in plaintext (even under encryption) corresponds to a prudent practice.

Definition 13 (plaintext) The set $plaintext(t)$ of *plaintext* of a term t is the set of extended names and variables that occurs in plaintext. It is recursively defined as follows.

$$\begin{array}{ll}
plaintext(u) = \{u\} & \text{if } u \text{ is a variable, an agent or a name} \\
plaintext(\mathrm{priv}(u)) = \{\mathrm{priv}(u)\} & \\
plaintext(\langle u_1, u_2\rangle) = plaintext(u_1) \cup plaintext(u_2) & \\
plaintext(\mathrm{h}(u)) = plaintext(u) & \\
plaintext(f(u_1, u_2)) = plaintext(u_1) & \text{for } f \in \{\mathrm{enc}, \mathrm{enca}, \mathrm{sign}\}
\end{array}$$

This notation is extended to set of terms and events. For protocols, we define $plaintext(\Pi)$ as follows:

$$plaintext(\Pi) = \{plaintext(\mathsf{e}) \mid \mathsf{e} \in \Pi \text{ and } \mathsf{e} \text{ is a communication event}\}.$$

Using our syntax, some protocols may still reveal critical keys in a hidden way. Consider for example the following one role (α-tagged) protocol.

$$\Pi(1) = \mathsf{snd}(\mathrm{enc}(\langle\alpha, a\rangle, k_{ab}));\ \mathsf{rcv}(\mathrm{enc}(\langle\alpha, a\rangle, x));\ \mathsf{snd}(x).$$

While the long-term key k_{ab} does not appear in plaintext, the key k_{ab} is revealed after simply one normal execution of the role. This protocol is however not realistic since it cannot be executed. Indeed, an unknown value cannot be learned (and sent) if it does not appear previously in plaintext. Thus we will further require (Condition 2 of Theorem 1) that a variable occurring in plaintext in a sent message, has to previously occur in plaintext in a received message.

5.3 Composition result

We show that two protocols can be safely composed as soon as they satisfy the disjoint encryption assumption and that critical long-term keys do not appear in plaintext. We also require that PS-LTL formulas also enjoy disjoint encryption with Π_2.

Theorem 1 (Main result) *Let* $\Pi_1 = [k_1] \to \mathsf{Roles}$, $\Pi_2 = [k_2] \to \mathsf{Roles}$ *be two protocols having disjoint encryption and such that* Π_2 *contains no status event. Let* T_0 *(intuitively the initial knowledge of the intruder) be a set of extended names. Let* $\mathsf{KC} = (\mathrm{n}(\Pi_1) \cup \mathrm{n}(\Pi_2)) \smallsetminus T_0$ *be the set of* critical extended names *and* ϕ *be a closed PS-*$\mathsf{LTL}^{\text{-}}$ *formula. Moreover, we assume that*

1. *critical extended names do not appear in plaintext, i.e.*
$$\mathsf{KC} \cap (plaintext(\Pi_1) \cup plaintext(\Pi_2)) = \emptyset.$$
2. *for any role* $\mathsf{e}_1, \ldots, \mathsf{e}_\ell$ *of* Π_1 *or* Π_2*, for any* i *such that* e_i *is a sent event, for any variable* $x \in plaintext(\mathsf{e}_i)$*, we have that* $x \in plaintext(\mathsf{e}_j)$ *for some new or received event* e_j *such that* $j < i$.
3. $EncSt(\phi)$ *and* $EncSt(\Pi_2)$ *have disjoint encryption.*

If $\Pi_1 \models \phi$ *for* T_0 *then* $\Pi_1 \mid \Pi_2 \models \phi$ *for* T_0.

We first discuss the hypotheses of the theorem. We have seen in Sections 5.1 and 5.2 that conditions 1 and 2 are necessary conditions. Note that condition 2 is actually satisfied by any realistic (executable) protocol since a party can send in plaintext only values that he knows already in plaintext. Condition 1 ensures that constant names that are not public do not appear in plaintext in Π_1 nor Π_2. This applies typically to the long-term private keys of protocols. These keys should indeed not be sent in plaintext. Note that this restriction does not apply to fresh keys or nonces generated during the execution of the protocols. Fresh keys and nonces are of course allowed to be sent in plaintext.

Condition 3 on the formula is not a real restriction since the security property should talk about protocol Π_1 thus if encrypted terms appear in the security property, they should be encrypted terms from Π_1, which have disjoint encryption with Π_2. We also require that Π_2 does not contain status event since we are interested to establish a security property on Π_1. It is a necessary condition as shown by the example below:

Example 6 Consider the two following 1-party protocols Π_1 and Π_2:

$$\Pi_1 = \mathsf{rcv}(x_1); \mathsf{event}(x_1); \mathsf{snd}(\mathrm{enc}(\langle \alpha, x_1 \rangle, k)) \qquad \Pi_2 = \mathsf{new}\ X; \mathsf{snd}(\mathrm{enc}(\langle \beta, X \rangle, k)).$$

Let $T_0 = \{\alpha, \beta\}$ and $\phi = \exists x.\mathsf{event}(\mathrm{enc}(\langle \beta, x \rangle, k))$. The conditions 1 and 2 stated in Theorem 1 are satisfied whereas condition 3 is not. We have that $\Pi_1 \mid \Pi_2 \models \phi$ for the initial knowledge T_0 whereas $\Pi_1 \not\models \phi$. Thus we have that $\neg\phi$ is a PS-$\mathsf{LTL}^{\text{-}}$ formula and $\Pi_1 \models \neg\phi$ while $\Pi_1 \mid \Pi_2 \not\models \neg\phi$.

We prove our combination result by contradiction and we first need to show that messages from two combined protocols do not need to be mixed up to mount an attack. For this purpose, we refine in Section 6 an existing decision procedure that allows us to control the form of the execution traces. Second, we show in Section 7 how to simplify the fragment of PS-LTL^{+} formula. Lastly, we provide a full proof of Theorem 1 in Section 8.

5.4 Applications

Security protocols can be analyzed using several existing tools, e.g. [10, 7]. The security of a protocol Π is however guaranteed provided that no other protocols share any of the private data of Π. Our result shows that, once the security of an isolated protocol has been established, this protocol can be safely executed in environments that may use some common data provided disjoint encryption is satisfied (and that long term private keys are not sent in plaintext). This condition is easy to check but might not be satisfied by existing protocols. A simple way to ensure it is to add the name of the protocol each time a party performs an encryption.

For security reasons however, most protocols actually make use of different keys. In this paper, we provide a simple criteria for safely composing protocols that share keys. This would allow to save both memory (for storing keys) and time since generating keys is time consuming in particular in the case of public key encryption. For example, the SSL protocol should contain the name "ssl2.0" in any of its encrypted messages. This would ensure that no armful interaction can occur with any other protocols even if they share some data with the SSL protocol, provided that these other protocols are also tagged. In other words, to avoid armful interaction between protocols, one should simply use a tagged version of them.

There are also situations were running different protocols with common keys already occur. We provide three examples of such cases.

- It is often the case that several versions of a protocol can be used concurrently. In this case and for backward compatibility reasons, the same keys can be used in the different versions of the protocol, which may lead to potentially dangerous interactions.
- When encrypting emails the same public key can be used for two distinct encryption protocols: the PGP protocol (Pretty Good Privacy) and its open source version OpenPGP. Note that the PGP protocol contains also other sub-protocols such as digitally signing message, all sharing the same public-key infrastructure.
- In the slightly different context of APIs, J. Clulow [17] discovered an attack when the VISA PIN verification values (PVV) protocol and the IBM CCA API share the same verification key.

6 Simplifying Constraint Systems

6.1 Constraint Systems

Constraint systems are quite common (see e.g. [38, 20, 23]) to model the execution of security protocols. We recall here their formalism.

Definition 14 (constraint system) A *constraint system* $\mathcal{C}$ is either $\perp$ or a finite sequence of expressions $(T_i \Vdash u_i)_{1 \leq i \leq n}$, called *constraints*, where each T_i is a finite set of terms, called the *left-hand side* of the constraint, and each u_i is a term, called the *right-hand side* of the constraint, such that:

- $\mathsf{init} \in T_1$ and $T_i \subseteq T_{i+1}$ for every i such that $1 \leq i < n$;
- if $x \in vars(T_i)$ then $\exists\, j < i$ such that $T_j = \min\{T \mid (T \Vdash u) \in \mathcal{C},\ x \in vars(u)\}$ (for the inclusion relation) and $T_j \subsetneq T_i$.

A *solution* of $\mathcal{C}$ is a closed substitution θ with $\mathrm{dom}(\theta) = \mathit{vars}(\mathcal{C})$ such that for every $(T \Vdash u) \in \mathcal{C}$, we have that $T\theta \vdash u\theta$. The empty constraint system is always satisfiable whereas $\perp$ denotes an unsatisfiable system.

A constraint system $\mathcal{C}$ is usually denoted as a conjunction of constraints $\mathcal{C} = \bigwedge_{1 \leq i \leq n}(T_i \Vdash u_i)$ with $T_i \subseteq T_{i+1}$, for all $1 \leq i < n$. The second condition in Definition 14 says that each time a new variable is introduced, it first occurs in some right-hand side. The left-hand side of a constraint system usually represents the messages sent on the network, while the right-hand side represents the message expected by the party.

Definition 15 Let Π be a protocol and sc be a scenario of Π. Let tr be a symbolic trace associated to Π and sc and T_0 be a finite set of terms. The constraint system $\mathcal{C}(\mathsf{tr})$ associated to tr and T_0 is defined as follows:

$$\mathcal{C}(\mathsf{tr}) := \{T_0 \cup \mathsf{K}(\mathsf{tr}_i) \Vdash u \mid \mathsf{tr}_i = \mathsf{tr}_{i-1};\, \mathsf{rcv}(u) \text{ and } 0 \leq i \leq \mathsf{length}(\mathsf{tr})\}.$$

Note that $\mathcal{C}(\mathsf{tr})$ satisfies the requirements given in Definition 14. In particular, the second condition is ensured thanks to the condition 1 of Definition 2. It is easy to establish the following result:

Lemma 1 *Let* tr *be a symbolic trace associated to a protocol* Π *and a scenario* sc. *Let* σ *be a substitution and* T_0 *be a finite set of terms. We have that*

$$\mathsf{tr}\sigma \textit{ is valid if and only if } \sigma \textit{ is a solution to } \mathcal{C}(\mathsf{tr}).$$

6.2 Simplification Rules

To prove our combination result, we first refine an existing decision procedure for solving constraint systems. Several decision procedures exist [36, 20, 23, 38, 16] for solving constraint systems. Some of them [36, 20, 23, 16] are based on a set of simplification rules allowing a general constraint system to be reduced to some simpler one, called *solved*, on which satisfiability can be easily decided. A constraint system is said *solved* [23] if it is different from $\perp$ and if each of its constraints is of the form $T \Vdash x$, where x is a variable. Note that the empty constraint system is solved. Solved constraint systems are particularly simple since they always have a solution. Indeed, let T_1 be the smallest (w.r.t. inclusion) left-hand side of a constraint. From the definition of a constraint system we have that $\mathsf{init} \in T_1$ and has no variable. Then the substitution τ defined by $x\tau = \mathsf{init}$ for every variable x is a solution since $T \vdash x\theta$ for any constraint $T \Vdash x$ of the solved constraint system. Given a constraint system $\mathcal{C}$, we say that T_i is a *minimal unsolved left-hand side* of $\mathcal{C}$ if T_i is a left-hand side of $\mathcal{C}$ and for all $T \Vdash u \in \mathcal{C}$ such that $T \subsetneq T_i$, we have that u is a variable.

The *simplification rules* we consider are given below. These are the simplification rules proposed in [23] except that we forbid unification between terms headed by pairs in the rules R_2 and R_3.

$$\mathsf{R}_1: \quad \mathcal{C} \wedge T \Vdash u \ \rightsquigarrow \ \mathcal{C} \qquad \text{if } T \cup \{x \mid T' \Vdash x \in \mathcal{C}, T' \subsetneq T\} \vdash u$$

$$\mathsf{R}_2: \quad \mathcal{C} \wedge T \Vdash u \rightsquigarrow_\sigma \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma \qquad \text{if } \sigma = \mathrm{mgu}(t, u) \text{ where } t \in St(T),\ t \neq u, \text{ and } t, u \text{ are neither variables nor pairs}$$

$$\mathsf{R}_3: \quad \mathcal{C} \wedge T \Vdash u \rightsquigarrow_\sigma \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma \qquad \text{if } \sigma = \mathrm{mgu}(t_1, t_2),\ t_1, t_2 \in St(T),\ t_1 \neq t_2, \text{ and } t_1, t_2 \text{ are neither variables nor pairs}$$

$$\mathsf{R}_4 \quad \mathcal{C} \wedge T \Vdash u \rightsquigarrow_\sigma \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma \quad \text{if } \sigma = \mathrm{mgu}(t_2, t_3) \text{ where } \mathrm{enca}(t_1, t_2) \in St(T), \mathrm{priv}(t_3) \in \mathit{plaintext}(T) \text{ and } t_2 \neq t_3$$

$$\mathsf{R}_5: \quad \mathcal{C} \wedge T \Vdash u \ \rightsquigarrow \ \bot \qquad \text{if } \mathit{vars}(T \cup \{u\}) = \emptyset \text{ and } T \not\vdash u$$

$$\mathsf{R}_6: \quad \mathcal{C} \wedge T \Vdash f(u_1, \ldots, u_n) \ \rightsquigarrow \ \mathcal{C} \wedge \{T \Vdash u_i \mid 1 \leq i \leq n\} \qquad \text{for } f \in \{\langle\rangle, \mathrm{enc}, \mathrm{enca}, \mathrm{sign}, \mathrm{h}\}$$

All the rules are indexed by a substitution (when there is no index then the identity substitution is assumed). We write $\mathcal{C} \rightsquigarrow_\sigma^* \mathcal{C}'$ if there are constraint systems $\mathcal{C}_1, \ldots, \mathcal{C}_n$ such that $\mathcal{C} \rightsquigarrow_{\sigma_0} \mathcal{C}_1 \rightsquigarrow_{\sigma_1} \ldots \rightsquigarrow_{\sigma_n} \mathcal{C}'$ and $\sigma = \sigma_0\sigma_1 \ldots \sigma_n$.

Since our rules are a subset of the rules of [23], our rules still transform a constraint system into a constraint system. Similarly, correction and termination are also ensured by [23]. It remains to show that they still form a complete decision procedure. This is formally stated in Theorem 2. Intuitively, unification between pairs is useless since pairs can be decomposed in order to perform unification on its components. Then, it is possible to build again the pair if necessary. Note that this is not always possible for encryption since the key used to decrypt or encrypt may be unknown by the attacker. Proving that forbidding unification between pairs still leads to a complete decision procedure required in particular to introduce a new notion of minimality for tree proofs for deduction. The fact that unification between pairs is useless has also been proved in [16] for another set of simplification rules.

Let $T_1 \subseteq T_2 \subseteq \ldots \subseteq T_n$. We say that a proof π of $T_i \vdash u$ is *left-minimal* if for any $j < i$ such that $T_j \vdash u$, π' is a proof of $T_j \vdash u$ where π' is obtained from π by replacing T_i with T_j in the left-hand side of each node of π.

Definition 16 (simple) We say that a proof π is *simple* if

1. any subproof of π is left-minimal,
2. a composition rule of the form $\dfrac{u_1 \quad u_2}{u}$ is not followed by a decomposition rule leading to u_1 or u_2,
3. any term of the form $\langle u_1, u_2 \rangle$ obtained by application of a decomposition rule or an axiom rule is directly followed by a projection rule.

Example 7 Let $T_1 = \{a\}$ and $T_2 = \{a, \mathrm{enc}(\langle a, b\rangle, k), k\}$. We have that $T_2 \vdash \langle a, b \rangle$.

$$\dfrac{\dfrac{}{T_2 \vdash \mathrm{enc}(\langle a, b\rangle, k)} \quad \dfrac{}{T_2 \vdash k}}{T_2 \vdash \langle a, b \rangle}$$

However, this proof is not a simple proof of $T_2 \vdash \langle a, b \rangle$. The term $\langle a, b \rangle$ has been obtained by an application of a decomposition rule. Thus we have to decompose it. A simple proof of $T_2 \vdash \langle a, b\rangle$ is described below:

$$\dfrac{\dfrac{}{T_2 \vdash a} \qquad \dfrac{\dfrac{\dfrac{}{T_2 \vdash \mathrm{enc}(\langle a, b\rangle, k)} \qquad \dfrac{}{T_2 \vdash k}}{T_2 \vdash \langle a, b\rangle}}{T_2 \vdash b}}{T_2 \vdash \langle a, b\rangle}$$

Then, we are able to prove completeness by relying on this notion of simple proof and on the following lemmas whose proofs are given in Appendix A. Our proof of Lemma 2 is similar to the one given in [23] with their own notion of simple proof (incomparable with the one we consider here). Nevertheless, we recall its proof in appendix for the sake of completeness. The proof of Lemma 3 is more involved and strongly relies on our notion of simple proof.

Lemma 2 *Let $\mathcal{C}$ be an unsolved constraint system, θ be a solution of $\mathcal{C}$ and $T_i \Vdash u_i$ be a minimal unsolved constraint of $\mathcal{C}$. Let u be a term. If there is a simple proof of $T_i\theta \vdash u$ having the last rule an axiom or a decomposition then there is $t \in St(T_i) \smallsetminus \mathcal{X}$ such that $t\theta = u$.*

Lemma 3 *Let $\mathcal{C}$ be an unsolved constraint system, θ be a solution of $\mathcal{C}$ and $T_i \Vdash v_i$ be a minimal unsolved constraint of $\mathcal{C}$ such that*

1. *for all $t_1, t_2 \in St(T_i)$ such that $t_1 \neq t_2$*

$$t_1\theta = t_2\theta \ \textit{ implies } \ t_1 \textit{ or } t_2 \textit{ is a variable or a pair}$$

2. *for all $\mathrm{enca}(t_1, t_2) \in St(T_i)$ and $\mathrm{priv}(t_3) \in plaintext(T_i)$*

$$t_2\theta = t_3\theta \textit{ implies } t_2 = t_3.$$

Assume $u_i \in St(T_i) \smallsetminus \mathcal{X}$ and $T_i\theta \vdash u_i\theta$. Then $T_i \cup \{x \mid T \Vdash x \in \mathcal{C}, T \subsetneq T_i\} \vdash u_i$.

Theorem 2 *Let $\mathcal{C}$ be an unsolved constraint system.*

1. *(Correctness) If $\mathcal{C} \rightsquigarrow^*_\sigma \mathcal{C}'$ for some constraint system $\mathcal{C}'$ and some substitution σ and if θ is a solution of $\mathcal{C}'$ then $\sigma\theta$ is a solution of $\mathcal{C}$.*
2. *(Completeness) If θ is a solution of $\mathcal{C}$, then there exist a solved constraint system $\mathcal{C}'$ and substitutions σ, θ' such that $\theta = \sigma\theta'$, $\mathcal{C} \rightsquigarrow^*_\sigma \mathcal{C}'$ and θ' is a solution of $\mathcal{C}'$.*
3. *(Termination) There is no infinite chain $\mathcal{C} \rightsquigarrow_{\sigma_1} \mathcal{C}_1 \ldots \rightsquigarrow_{\sigma_n} \mathcal{C}_n$.*

Proof Correction and termination are still ensured by [23]. Thus, we only have to show that the rules still form a complete decision procedure. Let $\mathcal{C}$ be an unsolved constraint system and θ be a solution of $\mathcal{C}$. We show that there is a constraint system $\mathcal{C}'$ and a solution τ of $\mathcal{C}'$ such that $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$ and $\theta = \sigma\tau$. Together with the termination property, this allows us to conclude that there exist a solved constraint system $\mathcal{C}''$ and substitutions σ', θ' such that $\theta = \sigma'\theta'$, $\mathcal{C} \rightsquigarrow^*_{\sigma'} \mathcal{C}''$ and θ' is a solution of $\mathcal{C}''$.

Consider the minimal unsolved constraint $T_i \Vdash u_i$ of $\mathcal{C}$. We have that u_i is not a variable whereas u_j is a variable for all $j < i$. Firstly, assume that $u_i = \langle v_1, v_2\rangle$ for some terms v_1, v_2. In such a case, let $\mathcal{C}'$ be the constraint system obtained from $\mathcal{C}$ by applying R_6 with $f = \langle\rangle$ and $\tau = \theta$. Since $T_i\theta \vdash u_i\theta$, we have also that $T_i\theta \vdash v_1\theta$ and $T_i\theta \vdash v_2\theta$ meaning that $\tau = \theta$ is a solution of $\mathcal{C}'$.

Now, assume that u_i is neither a variable nor a pair and consider a simple proof of $T_i\theta \vdash u_i\theta$. Depending on the last applied rule in this proof, we consider two cases.

1. The last rule is a composition.
   Suppose that it is the symmetric encryption rule. Hence, there are w_1, w_2 such that $T_i\theta \vdash w_1$ and $T_i\theta \vdash w_2$ and $\mathrm{enc}(w_1, w_2) = u_i\theta$. Since u_i is not a variable, there exist v_1, v_2 such that $u_i = \mathrm{enc}(v_1, v_2)$. Let $\mathcal{C}'$ be the constraint system obtained from $\mathcal{C}$ by applying the simplification rule R_6 on the constraint $T_i \Vdash \mathrm{enc}(v_1, v_2)$. Since $v_1\theta = w_1$ and $v_2\theta = w_2$, the substitution θ is also a solution to $\mathcal{C}'$. For the other composition rules the same reasoning holds, applying this time the corresponding R_6 rule.
2. The last rule is an axiom or a decomposition.
   Applying Lemma 2 we obtain that there is $t \in St(T_i) \smallsetminus \mathcal{X}$ such that $t\theta = u_i\theta$. We distinguish two cases:
   – $t \neq u_i$. Note that u_i is neither a pair nor a variable. Since $t\theta = u_i\theta$ we easily deduce that t is not a pair. Moreover t is not a variable, hence we can apply the simplification rule R_2.
   – $t = u_i$. In such a case, we have that $u_i \in St(T_i) \smallsetminus \mathcal{X}$. Either there are two distinct non variable and non pair terms $t_1, t_2 \in St(T_i)$ such that $t_1\theta = t_2\theta$ and we apply the simplification rule R_3. Or, there are $\mathrm{enca}(t_1, t_2) \in St(T_i)$, $\mathrm{priv}(t_3) \in \mathit{plaintext}(T_i)$ for some terms t_1, t_2, t_3 such that $t_2 \neq t_3$, and $t_2\theta = t_3\theta$. In such a case, we apply the simplification rule R_4. Otherwise, we deduce from Lemma 3 that the simplification rule R_1 can be applied. $\square$

Note that this result is of independent interest. Indeed, we provide a more efficient decision procedure for solving constraint systems, thus for deciding secrecy for a bounded number of sessions. Of course, the theoretical worst-case complexity remains the same (NP). Our complete set of simplification rules has been used in [6,5]. In [6], they improve existing decidability results in the context of verification protocols for an unbounded number of sessions. They bound the size of messages "for free" under a reasonable (syntactic) assumption on protocols. This condition is very similar to our notion of disjoint encryption. In [5], they propose a transformation which maps a protocol that is secure for a single session to a protocol that is secure for an unbounded number of sessions. Our set of simplification rules is used to establish the correctness of their transformation.

7 Simplifying PS-LTL Formulas

In order to establish our combination result for the PS-LTL formulas, we proceed in two steps. Following the approach of [21], we first show how to translate a closed PS-LTL$^+$ formula into an equivalent *elementary formula* (EF) (see Section 7.1) using the transformation **T** described in Section 7.2. Then, we will show in Section 8 how to prove our combination result for the corresponding fragment of the translated formulas.

7.1 Elementary Formulas

Definition 17 (Elementary formula) *Elementary formulas* EF are defined by the grammar:

$$\pi := \mathsf{true} \mid t_1 = t_2 \mid T \Vdash m \mid \neg\pi \mid \pi \vee \pi \mid \pi \wedge \pi \mid \exists x.\, \pi$$

where t_1, t_2 and m are terms, T is a finite set of terms and x is a variable.

The set of free variables of π, denoted by $\mathit{free}(\pi)$, is defined as usual. Sometimes, we write $t_1 \neq t_2$ instead of $\neg[t_1 = t_2]$.

Definition 18 Let π be an EF formula and σ be a closed substitution such that $\mathrm{dom}(\sigma) = \mathit{free}(\pi)$. Then $\sigma \models' \pi$ is defined inductively as follows:

$$
\begin{array}{lll}
\sigma \models' \mathsf{true} & & \\
\sigma \models' t_1 = t_2 & \text{iff} & t_1\sigma = t_2\sigma \\
\sigma \models' T \Vdash m & \text{iff} & T\sigma \vdash m\sigma \\
\sigma \models' \neg\pi & \text{iff} & \sigma \not\models' \pi \\
\sigma \models' \pi_1 \vee \pi_2 & \text{iff} & \sigma \models' \pi_1 \text{ or } \sigma \models' \pi_2 \\
\sigma \models' \pi_1 \wedge \pi_2 & \text{iff} & \sigma \models' \pi_1 \text{ and } \sigma \models' \pi_2 \\
\sigma \models' \exists x.\pi & \text{iff} & \exists t \in \mathsf{Terms} \text{ such that } t \text{ is ground and } \sigma \models' \pi[x \mapsto t]
\end{array}
$$

7.2 Translating PS-LTL^+ Formulas

We consider the fragment PS-LTL^+ that is made up of existential and positive PS-LTL formulas and we provide a translation in elementary formula for this fragment. Hence, we assume that ϕ is of the form $\exists\tilde{x}.\phi'$ where the formula ϕ' is quantifier-free. We define a translation $\mathbf{T}(\phi, \mathsf{tr}, T_0)$ from a PS-LTL^+ formula ϕ, a symbolic trace tr and an initial intruder knowledge T_0 into an EF formula. $\mathbf{T}(\phi, \mathsf{tr}, T_0)$ is the EF formula resulting from applying the transformation described below.

$$
\begin{array}{rcl}
\mathbf{T}(\mathsf{true}, \mathsf{tr}, T_0) & \rightarrow & \mathsf{true} \\
\mathbf{T}(\mathsf{learn}(m), \mathsf{tr}, T_0) & \rightarrow & T_0 \cup \mathsf{K}(\mathsf{tr}) \Vdash m \\
\mathbf{T}(\neg\phi, \mathsf{tr}, T_0) & \rightarrow & \neg\mathbf{T}(\phi, \mathsf{tr}, T_0) \\
\mathbf{T}(\phi_1 \wedge \phi_2, \mathsf{tr}, T_0) & \rightarrow & \mathbf{T}(\phi_1, \mathsf{tr}, T_0) \wedge \mathbf{T}(\phi_2, \mathsf{tr}, T_0) \\
\mathbf{T}(\phi_1 \vee \phi_2, \mathsf{tr}, T_0) & \rightarrow & \mathbf{T}(\phi_1, \mathsf{tr}, T_0) \vee \mathbf{T}(\phi_2, \mathsf{tr}, T_0) \\
\mathbf{T}(\exists x.\phi, \mathsf{tr}, T_0) & \rightarrow & \exists x.\mathbf{T}(\phi, \mathsf{tr}, T_0)
\end{array}
$$

For the temporal formulas, we first replace the 2^{nd} parameter tr by $\overline{\mathsf{tr}}$, i.e. we remove from tr all the communication and new events.

$$
\begin{array}{rl}
\mathbf{T}(P(t_1, \ldots, t_n), [\,], T_0) & \rightarrow \neg\mathsf{true} \\
\mathbf{T}(P(t_1, \ldots, t_n), \mathsf{tr};\, Q(t'_1, \ldots, t'_m), T_0) & \rightarrow \neg\mathsf{true} \quad \text{if } P \neq Q \text{ or } n \neq m \\
\mathbf{T}(P(t_1, \ldots, t_n), \mathsf{tr};\, P(t'_1, \ldots, t'_n), T_0) & \rightarrow t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \\
\mathbf{T}(\mathrm{Y}\psi, [\,], T_0) & \rightarrow \neg\mathsf{true} \\
\mathbf{T}(\mathrm{Y}\psi, \mathsf{tr}; \mathsf{e}, T_0) & \rightarrow \mathbf{T}(\psi, \mathsf{tr}, T_0) \\
\mathbf{T}(\psi_1 \,\mathrm{S}\, \psi_2, [\,], T_0) & \rightarrow \mathbf{T}(\psi_2, [\,], T_0) \\
\mathbf{T}(\psi_1 \,\mathrm{S}\, \psi_2, \mathsf{tr};\, \mathsf{e}, T_0) & \rightarrow \mathbf{T}(\psi_2, \mathsf{tr};\, \mathsf{e}, T_0) \vee \\
& \quad\ (\mathbf{T}(\psi_1, \mathsf{tr};\, \mathsf{e}, T_0) \wedge \mathbf{T}(\psi_1 \,\mathrm{S}\, \psi_2, \mathsf{tr}, T_0))
\end{array}
$$

The following lemma states that the translation $\mathbf{T}$ is correct, i.e. it preserves the semantics of PS-LTL^+ w.r.t. the semantics of EF .

Lemma 4 *Let ϕ be a closed PS-LTL^+ formula, tr be a (symbolic) trace, T_0 be a finite set of ground terms and σ be a closed substitution with $\mathit{vars}(\mathsf{tr}) = \mathrm{dom}(\sigma)$. Then we have that*

$$\langle \mathsf{tr}\sigma, T_0 \rangle \models \phi \ \textit{ if and only if } \ \sigma \models' \mathbf{T}(\phi, \mathsf{tr}, T_0).$$

Moreover, atomic formula of the form $T \Vdash m$ occurs positively in $\mathbf{T}(\phi, \mathsf{tr}, T_0)$, i.e. any occurrence of $T \Vdash m$ in $\mathbf{T}(\phi, \mathsf{tr}, T_0)$ appears under an even number of negation.

The proof can be easily done by induction on the number of rewriting steps to obtain the EF formula associated to $\mathbf{T}(\phi, \mathsf{tr}, T_0)$. This has been done in [21] in a rather similar setting.

8 Proof of our combination result

This section is devoted to the proof of Theorem 1. The proof is done in three main steps. First, Theorem 2 serves as a key result for proving that if there exists a substitution σ such that $\mathsf{tr}\sigma$ is valid and $\langle \mathsf{tr}\sigma, T_0 \rangle \models \phi$, then there exists one, say θ, where messages from Π_1 and Π_2 are not mixed up. Second, conditions 1-3 allow us to control the position of the critical extended names KC: those names do not occur in plaintext position. This is the purpose of Section 8.1. Third, thanks to the two previous steps, we prove that terms issued from Π_2 are not useful for deducing terms issued from Π_1. This is formally stated and proved in Section 8.2. In Section 8.3, we complete the proof of Theorem 1.

8.1 Existence of a solution without any mixing

In this subsection, we show that when there exists a solution, there is one, say θ, satisfying some particular conditions (see Proposition 2). First of all, messages from Π_1 and Π_2 are not mixed-up. This is obtained by observing that the simplification rules enable us to build θ step by step through unification of subterms of Π_1 and Π_2. Now, since unification between pairs is forbidden, the rules R_2 and R_3 only involve subterms issued from the same protocol (thanks to the disjoint encryption hypothesis). Second, conditions 1-3 allow us to control the position of the critical extended names KC.

The *left-hand side* of a constraint system $\mathcal{C}$, denoted by $\mathsf{lhs}(\mathcal{C})$, is the maximal left-hand side of the constraints of $\mathcal{C}$. The *right-hand side* of a constraint system $\mathcal{C}$, denoted by $\mathsf{rhs}(\mathcal{C})$, is the set of right-hand sides of its constraints.

Definition 19 (well-formed) Let T be a set of terms and KC be a set of extended names. A constraint system $\mathcal{C}$ is *well-formed* w.r.t. T and KC if

- $\mathsf{lhs}(\mathcal{C}) \cup \mathsf{rhs}(\mathcal{C}) \subseteq T$,
- the constraint system $\mathcal{C}$ satisfies the *plaintext origination property*, that is if $x \in \mathit{plaintext}(T') \cap \mathcal{X}$ for some $(T' \Vdash u') \in \mathcal{C}$ then

  $$T^p_x \overset{\mathsf{def}}{=} \min\{T'' \mid (T'' \Vdash u'') \in \mathcal{C} \text{ and } x \in \mathit{plaintext}(u'')\}$$

  exists and $T^p_x \subsetneq T'$.
- $\mathsf{KC} \cap \mathit{plaintext}(\mathsf{lhs}(\mathcal{C})) = \emptyset$.

Lemma 5 *Let T_1 and T_2 be two sets of terms having disjoint encryption and* KC *be a set of extended names. Let $\mathcal{C}$ be a well-formed constraint system w.r.t. $St(T_1) \cup St(T_2)$ and* KC*. Let $\mathcal{C}'$ and σ be such that $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$ with $\mathcal{C}'$ satisfiable. Then, we have that*

1. *$T_1\sigma$ and $T_2\sigma$ have disjoint encryption,*
2. *$names(T_i\sigma) \subseteq names(T_i)$ for $i = 1, 2$, and*
3. *the constraint system $\mathcal{C}'$ is well-formed w.r.t. $St(T_1\sigma) \cup St(T_2\sigma)$ and* KC.

We define the set Sinit of terms of the form $\langle \mathsf{init}, \langle \mathsf{init} \ldots \rangle\rangle$. Formally, Sinit is the smallest set such that $\mathsf{init} \in \mathsf{Sinit}$ and for any $t \in \mathsf{Sinit}$, $\langle \mathsf{init}, t\rangle \in \mathsf{Sinit}$.

Lemma 6 *Let $\mathcal{C}$ be a constraint system in solved form and* DEq *be a finite set of disequations such that τ is a solution of $\mathcal{C} \wedge \mathsf{DEq}$. There exists a solution τ' of $\mathcal{C} \wedge \mathsf{DEq}$ such that for every variable $x \in \mathrm{dom}(\tau')$, we have that $x\tau' \in \mathsf{Sinit}$.*

The proof of the two lemmas above can be found in Appendix B.1.

Proposition 2 *Let T_0 and* KC *be two sets of extended names. Let T_1 and T_2 be two sets of terms having disjoint encryption and $\mathcal{C}$ be a well-formed constraint system w.r.t. $T_0 \cup T_1 \cup T_2$ and* KC*. Let* DEq *be a finite set of disequations and* Eq *be a finite set of equations such that $\{t_1, t_2 \mid t_1 = t_2 \in \mathsf{Eq}\} \subseteq T_1$. Let θ be a solution of $\mathcal{C} \wedge \mathsf{Eq} \wedge \mathsf{DEq}$. There exists a solution θ' of $\mathcal{C} \wedge \mathsf{Eq} \wedge \mathsf{DEq}$ such that*

1. *$T_1\theta'$ and $T_2\theta'$ have disjoint encryption, and*
2. *$names(T_i\theta') \subseteq names(T_i)$, for $i = 1, 2$.*

Proof Let T_0, T_1, T_2, KC, $\mathcal{C}$, DEq, Eq and θ as explained above. Let ρ and σ be two substitutions such that $\theta = \rho\sigma$ and $\rho = \mathrm{mgu}(\mathsf{Eq})$. Thanks to our completeness result (Theorem 2), we know that there exists a constraint system $\mathcal{C}'$ in solved form and a substitution σ' such that $\mathcal{C}\rho \rightsquigarrow^*_{\sigma'} \mathcal{C}'$. Moreover, we know that there exists τ solution of $\mathcal{C}'$ such that $\sigma = \sigma'\tau$. The substitution τ is also a solution of $\mathsf{DEq}\rho\sigma'$. Hence, by applying Lemma 6, we know that there exists a solution τ' of $\mathcal{C}' \wedge \mathsf{DEq}\rho\sigma'$ such that $x\tau'$ is a pair of init for any $x \in \mathrm{dom}(\tau')$. Let $\theta' = \rho\sigma'\tau'$. By construction, we have that θ' is a solution of $\mathcal{C} \wedge \mathsf{Eq} \wedge \mathsf{DEq}$. It remains to show the two points stated in the proposition.

By hypothesis, the sets T_1 and T_2 have disjoint encryption. Since we have that $\{t_1, t_2 \mid t_1 = t_2 \in \mathsf{Eq}\} \subseteq T_1$, we can easily show (by relying on the unification algorithm given in [35]) that $St(T_1\rho) \subseteq (St(T_1) \smallsetminus \mathcal{X})\rho$. Thus, we have that $T_1\rho$ and $T_2\rho = T_2$ have disjoint encryption. Then, thanks to Lemma 5, we obtain that:

- $T_1\rho\sigma'$ and $T_2\rho\sigma' = T_2\sigma'$ have disjoint encryption,
- $names(T_i\rho\sigma') \subseteq names(T_i\rho) \subseteq names(T_i)$ for $i = 1, 2$.

From these facts, we easily deduce that $T_1\theta'$ and $T_2\theta'$ have disjoint encryption and also that $names(T_i\theta') \subseteq names(T_i)$ for $i = 1, 2$. $\square$

8.2 Getting rid of the terms coming from Π_2

In this subsection, we prove that terms issued from Π_2 are not useful for deducing terms issued from Π_1. For this, we establish that $T \vdash u$ implies $\overline{T} \vdash \overline{u}$ where $\overline{\cdot}$ is a function that keep the terms issued from Π_1 unchanged and projects the terms issued from Π_2 on the special constant init. The proof is done by induction on the proof witnessing $T \vdash u$. It requires in particular the introduction of a new locality lemma for deduction of ground terms (Lemma 7).

Given a set Names of names ($\mathsf{Names} \subseteq \mathcal{N}$) and a set ETerms of terms, we define the function $\overline{\cdot}$ inductively as follows:

- $\overline{u} = \mathsf{init}$ if $u \in \mathsf{Names}$,
- $\overline{u} = u$ if u is an agent, or a name such that $u \notin \mathsf{Names}$,
- $\overline{f(u_1, \ldots, u_n)} = \mathsf{init}$ if $f(u_1, \ldots, u_n) \in EncSt(\mathsf{ETerms})$
- $\overline{f(u_1, \ldots, u_n)} = f(\overline{u_1}, \ldots, \overline{u_n})$ otherwise

In the remaining we assume given a set of names Names and a set of terms ETerms. The function $\overline{\cdot}$ is defined w.r.t. to these two sets. Intuitively, Names will be the fresh names introduced by Π_2 and ETerms will be the encrypted terms introduced by Π_2. Thanks to the disjoint encryption property, these terms will be disjoint from the terms coming from Π_1.

Our locality lemma relies on the following definition. The proofs of Lemmas 7 and 8 can be found in Appendix B.2.

Definition 20 **($St_{plain}(t)$)** Let t be a ground term. The set $St_{plain}(t)$ of subterms of t that appear at a plaintext position is inductively defined as follows:

- $St_{plain}(u) = \{u\}$ if u is an extended name
- $St_{plain}(f(u_1, u_2)) = \{f(u_1, u_2)\} \cup St_{plain}(u_1)$ if $f \in \{\mathrm{enc}, \mathrm{enca}, \mathrm{sign}\}$
- $St_{plain}(\mathrm{h}(u)) = \{\mathrm{h}(u)\} \cup St_{plain}(u)$
- $St_{plain}(\langle u_1, u_2 \rangle) = \{\langle u_1, u_2 \rangle\} \cup St_{plain}(u_1) \cup St_{plain}(u_2)$.

Lemma 7 (locality) *Let T be a set of terms and u be a term such that $T \vdash u$. Let π be a proof of $T \vdash u$ which is minimal w.r.t. its number of nodes. Then π only involves terms in $St(T \cup \{u\})$. Moreover, if π ends with a decomposition rule or the axiom rule then π only involves terms in $St(T)$ and $u \in St_{plain}(T)$.*

Lemma 8 *Let T_0 be a set of terms such that $names(T_0) \cap \mathsf{Names} = \emptyset$ and $\mathsf{init} \in T_0$. Let v be a term such that $plaintext(v) \subseteq T_0 \cup \mathsf{Names}$ and $EncSt(v) \subseteq EncSt(\mathsf{ETerms})$. Then, we have that $T_0 \vdash \overline{v}$.*

Proposition 3 *Let T_0 be a set of extended names such that $names(T_0) \cap \mathsf{Names} = \emptyset$ and $\mathsf{init} \in T_0$. Let T_1 and T_2 be two sets of terms such that:*

- *$names(T_1) \cap \mathsf{Names} = \emptyset$ and $EncSt(T_1) \cap EncSt(\mathsf{ETerms}) = \emptyset$,*
- *$plaintext(T_2) \subseteq T_0 \cup \mathsf{Names}$ and $EncSt(T_2) \subseteq EncSt(\mathsf{ETerms})$.*

Let u be a term such that $T_0, T_1, T_2 \vdash u$. We have also that $T_0, T_1 \vdash \overline{u}$.

Proof We first establish that $\overline{T_0}, \overline{T_1}, \overline{T_2} \vdash \overline{u}$. Let π be a proof of $T_0, T_1, T_2 \vdash u$ which is minimal *w.r.t.* its number of nodes. We will show that $\overline{T_0}, \overline{T_1}, \overline{T_2} \vdash \overline{u}$ by induction on the proof, depending on the last rule that has been applied.

- If the last rule is an axiom. In such a case, we have that $u \in T_0 \cup T_1 \cup T_2$. We easily deduce that $\overline{u} \in \overline{T_0} \cup \overline{T_1} \cup \overline{T_2}$. This allows us to conclude.

– If the last rule is a composition. Either $\overline{u} = \mathsf{init}$ and we easily conclude. Otherwise, suppose for example that the last rule is the symmetric decryption rule. In such a case, we have that $u = \mathrm{enc}(u_1, u_2)$ and $\overline{u} = \mathrm{enc}(\overline{u_1}, \overline{u_2})$. By induction hypothesis, we know that $\overline{T_0}, \overline{T_1}, \overline{T_2} \vdash \overline{u_1}$ and $\overline{T_0}, \overline{T_1}, \overline{T_2} \vdash \overline{u_2}$. Hence, we deduce that $\overline{T_0}, \overline{T_1}, \overline{T_2} \vdash \mathrm{enc}(\overline{u_1}, \overline{u_2})$, that is $\overline{T_0}, \overline{T_1}, \overline{T_2} \vdash \overline{u}$.
– If the last rule is a decomposition, for example the symmetric decryption rule. In such a case, we have that

$$\frac{\pi_1 = \left\{ \frac{\cdots}{T_0, T_1, T_2 \vdash \mathrm{enc}(u, v)} \right. \qquad \pi_2 = \left\{ \frac{\cdots}{T_0, T_1, T_2 \vdash v} \right.}{T_0, T_1, T_2 \vdash u}$$

If $\mathrm{enc}(u, v) \notin \mathit{EncSt}(\mathsf{ETerms})$, then by applying our induction hypothesis, we easily conclude since $\overline{\mathrm{enc}(u, v)} = \mathrm{enc}(\overline{u}, \overline{v})$. Now, we have to consider the case where $\mathrm{enc}(u, v) \in \mathit{EncSt}(\mathsf{ETerms})$, i.e. $\overline{\mathrm{enc}(u, v)} = \mathsf{init}$. By minimality of the proof we know that π_1 ends either with an axiom rule or with a decomposition rule. Hence, we have that $\mathrm{enc}(u, v) \in \mathit{St}_{plain}(T_0 \cup T_1 \cup T_2)$ thanks to Lemma 7. Since $\mathrm{enc}(u, v) \in \mathit{EncSt}(\mathsf{ETerms})$ and $\mathit{EncSt}(T_1) \cap \mathit{EncSt}(\mathsf{ETerms}) = \emptyset$, we deduce that $\mathrm{enc}(u, v) \in \mathit{St}_{plain}(T_2)$, thus $u \in \mathit{St}_{plain}(T_2)$. Since $\mathit{plaintext}(T_2) \subseteq T_0 \cup \mathsf{Names}$, we deduce that $\mathit{plaintext}(u) \subseteq T_0 \cup \mathsf{Names}$. Since $\mathrm{enc}(u, v) \in \mathit{EncSt}(\mathsf{ETerms})$, we also have that $\mathit{EncSt}(u) \subseteq \mathit{EncSt}(\mathsf{ETerms})$. Lemma 8 allows us to conclude that $T_0 \vdash \overline{u}$. For the asymmetric decryption rule and the optional signature rule, a similar reasoning holds. For the projection rules, the reasoning is even easier since we have $\overline{\langle u_1, u_2 \rangle} = \langle \overline{u_1}, \overline{u_2} \rangle$ thus we can always applied the induction hypothesis.

Hence, we have shown that $\overline{T_0}, \overline{T_1}, \overline{T_2} \vdash \overline{u}$. By hypothesis, we know that T_0 is a set of extended names such that $\mathit{names}(T_0) \cap \mathsf{Names} = \emptyset$. Thus, we easily deduce that $\overline{T_0} = T_0$. By hypothesis, we have that $\mathit{names}(T_1) \cap \mathsf{Names} = \emptyset$ and $\mathit{EncSt}(T_1) \cap \mathit{EncSt}(\mathsf{ETerms}) = \emptyset$. Thus, we have that $\overline{T_1} = T_1$. Now, by applying Lemma 8 on each term $v \in T_2$, we easily obtain that $T_0 \vdash \overline{v}$. From all these facts, we easily deduce that $T_0, T_1 \vdash \overline{u}$. □

8.3 Proof of Theorem 1

Our main composition result relies on the following proposition, which relates the traces of $\Pi_1 \mid \Pi_2$ with the traces of Π_1.

Proposition 4 *Let $\Pi_1 = [k_1] \to \mathsf{Roles}$ and $\Pi_2 = [k_2] \to \mathsf{Roles}$ be two protocols having disjoint encryption and such that Π_2 contains no status event. Let T_0 (intuitively the initial knowledge of the intruder) be a set of extended names. Let $\mathsf{KC} = (\mathrm{n}(\Pi_1) \cup \mathrm{n}(\Pi_2)) \smallsetminus T_0$ be the set of* critical extended names *and ϕ be a closed PS-LTL^+ formula. Moreover, we assume that*

1. *critical extended names do not appear in plaintext, i.e.*

$$\mathsf{KC} \cap (\mathit{plaintext}(\Pi_1) \cup \mathit{plaintext}(\Pi_2)) = \emptyset.$$

2. *for any role $\mathsf{e}_1, \ldots, \mathsf{e}_\ell$ of Π_1 or Π_2, for any i such that e_i is a sent event, for any variable $x \in \mathit{plaintext}(\mathsf{e}_i)$, we have that $x \in \mathit{plaintext}(\mathsf{e}_j)$ for some new or received event e_j such that $j < i$.*

3. *EncSt*(ϕ) *and* *EncSt*(Π_2) *have disjoint encryption.*

Let $k = k_1 + k_2$ *and* sc *be a scenario for* $\Pi_1 \mid \Pi_2$. *Let* tr *be the symbolic trace associated to* sc *and* T_0. *Let* $\mathsf{sc}' = \mathsf{sc}|_{\Pi_1}$ *and* tr' *be the symbolic trace associated to* sc' *and* T_0. *If there exists* σ *such that* $\mathsf{tr}\sigma$ *is valid and* $\langle \mathsf{tr}\sigma, T_0\rangle \models \phi$ *then there exists* σ' *such that* $\mathsf{tr}'\sigma'$ *is valid and* $\langle \mathsf{tr}'\sigma', T_0\rangle \models \phi$.

Proof Let $\Pi_1 : [k_1] \to \mathsf{Roles}$, $\Pi_2 : [k_2] \to \mathsf{Roles}$, T_0 and ϕ defined as in Proposition 4. Let $k = k_1 + k_2$ and sc be a scenario for $\Pi_1 \mid \Pi_2$. Let tr be the symbolic trace associated to $\Pi_1 \mid \Pi_2$ and sc. Let $\mathsf{sc}' := \mathsf{sc}|_{\Pi_1}$ and tr' be the symbolic trace associated to Π_1 and sc'. Since ϕ is a PS-LTL^+ formula, we have that ϕ is of the form $\exists\tilde{x}.\phi_0$ for some PS-LTL^+ formula ϕ_0 without any quantifier.

Let σ be a substitution such that $\mathsf{tr}\sigma$ is valid and $\langle \mathsf{tr}\sigma, T_0\rangle \models \phi$. Thus, thanks to Lemma 4, we have that $\sigma \models' \mathbf{T}(\phi, \mathsf{tr}, T_0)$. We have that $\mathbf{T}(\phi, \mathsf{tr}, T_0) = \exists\tilde{x}.\psi_0$ for some EF formula ψ_0 without any quantifier. Moreover, thanks to Lemma 4 we have that atomic formulas of the form $T \Vdash m$ appear under an even number of negations. We transform ψ_0 into its disjunctive normal form, thus $\psi_0 = \bigvee_{1\leq j\leq \ell} \psi_j$ We know that there exists j such that $\mathsf{tr}\sigma$ is valid and $\sigma \models' \exists\tilde{x}.\psi_j$. Moreover, the EF formula ψ_j can be written as $\mathsf{Ded} \wedge \mathsf{Eq} \wedge \mathsf{DEq}$ where:

- Ded is a finite set of deduction constraints of the form $T_0 \cup \mathsf{K}(\mathsf{tr}) \Vdash m$ for some term m,
- Eq (resp. DEq) is a finite set of equations (resp. disequations) of the form $t_1 = t_2$ (resp. $t_1 \neq t_2$) where $t_1 \in St(\phi)$ and $t_2 \in St(\mathsf{e})$ for some $\mathsf{e} \in \mathsf{tr}$.

We assume that the variables $\tilde{x}$ do not occur in tr. Thus, we have that:

- σ is a solution of $\mathcal{C} := C(\mathsf{tr}); \mathsf{Ded}$. (Lemma 1). Note also that $\mathcal{C}$ is a constraint system which satisfies the plaintext origination property. This is due to the fact that the protocols we consider satisfy condition 2 (stated in Proposition 4).
- $t_1\sigma = t_2\sigma$ for every $t_1 = t_2 \in \mathsf{Eq}$,
- $t_1\sigma \neq t_2\sigma$ for every $t_1 \neq t_2 \in \mathsf{DEq}$.

Let $\mathcal{C}' = \mathcal{C}(\mathsf{tr}'); \mathsf{Ded}'$ where $\mathsf{Ded}' = \{(T_0 \cup \mathsf{K}(\mathsf{tr}') \Vdash m) \mid (T_0 \cup \mathsf{K}(\mathsf{tr}) \Vdash m) \in \mathsf{Ded}\}$. We have to show that $\mathcal{C}' \wedge \mathsf{Eq} \wedge \mathsf{DEq}$ has a solution which would mean that Π_1 does satisfy $\exists\tilde{x}.\psi_j$, and thus Π_1 satisfies ϕ.

The constraint systems $\mathcal{C}$ and $\mathcal{C}'$ are as follows:

$$\mathcal{C} := \begin{cases} T_0 & \Vdash\ u_1 \\ T_0, v_1 & \Vdash\ u_2 \\ T_0, v_1, v_2 & \Vdash\ u_3 \\ \cdots & \Vdash\ \cdots \\ T_0, v_1, \ldots, v_n & \Vdash\ m_1 \\ \cdots & \Vdash\ \cdots \\ T_0, v_1, \ldots, v_n & \Vdash\ m_k \end{cases} \qquad \mathcal{C}' := \begin{cases} T_0 & \Vdash\ u_{i_1} \\ T_0, v_{i_1} & \Vdash\ u_{i_2} \\ T_0, v_{i_1}, v_{i_2} & \Vdash\ u_{i_3} \\ \cdots & \Vdash\ \cdots \\ T_0, v_{i_1}, \ldots, v_{i_n} & \Vdash\ m_1 \\ \cdots & \Vdash\ \cdots \\ T_0, v_{i_1}, \ldots, v_{i_n} & \Vdash\ m_k \end{cases}$$

where $i_1, \ldots, i_n$ is a sequence obtained from $1 \ldots n$ by removing the elements corresponding to a step of the protocol Π_2. The k last deduction constraints correspond to those in Ded (resp. Ded').

Before applying Proposition 2, we have to check that all the hypotheses are satisfied. Let

- $T_1 = \{u_{i_1}, v_{i_1}, \dots, u_{i_n}, v_{i_n}, m_1, \dots, m_k\} \cup \{t_1, t_2 \mid t_1 = t_2 \in \mathsf{Eq}\}$
- $T_2 = \{u_j, v_j \mid 1 \leq j \leq n \text{ and } j \notin \{i_1, \dots, i_n\}\}$.

First of all, we have that T_1 and T_2 are two sets of terms having disjoint encryption. This is because terms in T_1 come from Π_1 and ϕ whereas terms in T_2 come from Π_2. We have also that $\mathcal{C}$ is well-formed w.r.t. $T_0 \cup T_1 \cup T_2$ and KC. Hence, we apply Proposition 2 in order to deduce that there exists a solution θ of $\mathcal{C} \wedge \mathsf{Eq} \wedge \mathsf{DEq}$ and such that:

1. $T_1\theta$ and $T_2\theta$ have disjoint encryption, and
2. $names(T_i\theta) \subseteq names(T_i)$ for $i = 1, 2$.

Let $\theta' = \theta|_V$ where V is the set of variables which appear in $\mathcal{C}' \wedge \mathsf{Eq} \wedge \mathsf{DEq}$. To conclude, it remains to show that θ' is a solution of $\mathcal{C}'$.

Let $\mathsf{Names} = \{\mathrm{img}(\sigma_{r,s}) \cap \mathcal{N} \mid (r, s) \in \mathsf{sc} \text{ and } r > k_1\}$, i.e. all the names generating during the execution of Π_2 and $\mathsf{ETerms} = T_2\theta$. We have that $\mathsf{Names} \cap names(T_0) = \emptyset$ and $\mathsf{Names} \cap \mathsf{KC} = \emptyset$. Note also that $EncSt(T_1\theta) \cap EncSt(\mathsf{ETerms}) = \emptyset$ since $T_1\theta$ and $T_2\theta$ have disjoint encryption.

Let $T \vdash u$ be a constraint in $\mathcal{C}$. Either the corresponding constraint has been removed in $\mathcal{C}'$. Otherwise, we have that $T = T_0 \cup \{v_1, \dots, v_j\}$ for some j and the corresponding constraint in $\mathcal{C}'$ is $T' \vdash u$ where $T' = T_0 \cup \{v_{i_1}, \dots, v_{i_j}\}$. Moreover, in such a case, we have that $u \in T_1$, and thus $u\theta \in T_1\theta$. Thanks to the fact that θ is a solution of $\mathcal{C}$, we know that: $T_0, v_1\theta, v_2\theta, \dots, v_j\theta \vdash u\theta$. Thanks to Proposition 3, we obtain that $T_0, v_{i_1}\theta, \dots, v_{i_j}\theta \vdash \overline{u\theta}$, i.e. $T'\theta' \vdash u\theta'$ since $\overline{u\theta} = u\theta$ and $\theta' = \theta|_{vars(\mathcal{C}')}$. $\square$

We are now ready to complete the proof of Theorem 1.

Proof Assume by contradiction that $\Pi_1 \mid \Pi_2 \not\models \phi$ for the initial knowledge T_0. It means that there exists a scenario sc for which the symbolic trace tr associated to $\Pi_1 \mid \Pi_2$ and sc satisfies the following requirement:

there exists a substitution σ such that $\mathsf{tr}\sigma$ is valid and $\langle \mathsf{tr}\sigma, T_0 \rangle \models \neg\phi$.

Let $\mathsf{sc}' = \mathsf{sc}|_{\Pi_1}$ and tr' be the symbolic trace associated to Π_1 and sc. Thanks to Proposition 4 (note that $\neg\phi$ is a PS-LTL^+ formula), we easily deduce that there exists σ' such that $\mathsf{tr}'\sigma'$ is valid and $\langle \mathsf{tr}'\sigma', T_0 \rangle \models \neg\phi$. This means that $\Pi_1 \not\models \phi$, thus a contradiction. $\square$

9 Conclusion

In this paper, we have shown that secure protocols can be safely executed in the presence of other protocols, as soon as encrypted sub-messages from different messages are not unifiable. This can be easily achieved by tagging protocols, that is, adding a protocol identifier in each encrypted message. Our result holds for a large class of security properties that encompasses secrecy and various formulations of authenticity.

We foresee composition results in a more general way. In this paper, protocols are composed in the sense that they can be executed in the same environment. We plan to develop composition results where protocols can use other protocols as sub-programs. For example, a protocol could use a secure channel, letting the implementation of the secure channel underspecified. This secure channel could be then possibly implemented by any protocol establishing session keys.

References

1. M. Abadi and R. M. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Trans. Software Eng.*, 22(1):6–15, 1996.
2. M. Abadi and P. Rogaway. Reconciling two views of cryptography. In *Proc. of the International Conference on Theoretical Computer Science (IFIP TCS2000)*, pages 3–22, August 2000.
3. R. Amadio and W. Charatonik. On name generation and set-based analysis in the Dolev-Yao model. In *Proc. International Conference on Concurrency Theory (CONCUR'02)*, volume 2421 of *LNCS*, pages 499–514. Springer-Verlag, 2002.
4. S. Andova, C. Cremers, K. G. Steen, S. Mauw, S. M. lsnes, and S. Radomirović. Sufficient conditions for composing security protocols. *Information and Computation*, 206(2-4):425–459, 2008.
5. M. Arapinis, S. Delaune, and S. Kremer. From one session to many: Dynamic tags for security protocols. In I. Cervesato, editor, *Proc. 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'08)*, LNAI, Doha, Qatar, 2008. Springer. To appear.
6. M. Arapinis and M. Duflot. Bounding messages for free in security protocols. In *Proc. 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *LNCS*, pages 376–387. Springer, 2007.
7. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The Avispa tool for the automated validation of internet security protocols and ap plications. In *Proc. 17th International Conference on Computer Aided Verification (CAV'05)*, volume 3576 of *LNCS*, 2005.
8. M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations (extended abstract). In *Proc. of 10th ACM Conference on Computer and Communications Security (CCS'05)*, pages 220 – 230, 2003.
9. B. Barak, R. Canetti, J. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *Proc. 45th Symposium on Foundations of Computer Science (FOCS'04)*, pages 186–195. IEEE Comp. Soc. Press, 2004.
10. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Comp. Soc. Press, 2001.
11. B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In *Proc. 6th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *LNCS*. Springer, 2003.
12. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd Annual Symposium on Foundations of Computer Science (FOCS'01)*, pages 136–145. IEEE Comp. Soc., 2001.
13. R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In *Proc. 4th Theory of Cryptography Conference, (TCC'07)*, LNCS, pages 61–85. Springer, 2007.
14. R. Canetti, C. Meadows, and P. F. Syverson. Environmental requirements for authentication protocols. In *Proc. Symposium on Software Security – Theories and Systems*, volume 2609 of *LNCS*, pages 339–355. Springer, 2002.
15. R. Canetti and T. Rabin. Universal composition with joint state. In *Proc. 23rd International Cryptology Conference (CRYPTO'03)*, LNCS, pages 265–281. Springer, 2003.
16. Y. Chevalier. *Résolution de problèmes d'accessibilité pour la compilation et la validation de protocoles cryptographiques*. PhD thesis, Université Henri Poincaré, Nancy (France), 2003.
17. J. Clulow. The design and analysis of cryptographic APIs for security devices. Master's thesis, University of Natal, Durban, South Africa, 2003. Chapter 3.
18. H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proc. 14th Int. Conf. on Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *LNCS*, pages 148–164. Springer-Verlag, June 2003.
19. H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. *Science of Computer Programming*, 50(1-3):51–71, 2004.

20. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. 18th Annual Symposium on Logic in Comp. Sc. (LICS'03)*, pages 271–280. IEEE Comp. Soc. Press, 2003.
21. R. Corin. *Analysis Models for Security Protocols*. PhD thesis, University of Twente, 2006.
22. V. Cortier, J. Delaitre, and S. Delaune. Safely composing security protocols. In *Proc. 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *LNCS*, pages 352–363. Springer, 2007.
23. V. Cortier and E. Zalinescu. Deciding key cycles for security protocols. In *Proc. 13th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06)*, volume 4246 of *LNCS*, pages 317–331. Springer, 2006.
24. C. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Ph.D. dissertation, Eindhoven University of Technology, 2006.
25. A. Datta, A. Derek, J. C. Mitchell, and A. Roy. Protocol composition logic (PCL). *Electr. Notes Theoretical Computer Science*, 172:311–358, 2007.
26. S. Delaune, S. Kremer, and M. D. Ryan. Composition of password-based protocols. In *Proc. of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 239–251, Pittsburgh, PA, USA, 2008. IEEE Computer Society Press.
27. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. of the Workshop on Formal Methods and Security Protocols*, 1999.
28. L. Gong and P. Syverson. Fail-stop protocols: An approach to designing secure protocols. In *Proc. 5th International Working Conference on Dependable Computing for Critical Applications*, pages 44–55, 1995.
29. J. D. Guttman and F. J. Thayer. Protocol independence through disjoint encryption. In *Proc. 13th Computer Security Foundations Workshop (CSFW'00)*, pages 24–34. IEEE Comp. Soc. Press, 2000.
30. J. Kelsey, B. Schneier, and D. Wagner. Protocol interactions and the chosen protocol attack. In *Proc. 5th International Workshop on Security Protocols*, volume 1361 of *LNCS*, pages 91–104. Springer, 1997.
31. R. Küsters and M. Tuengerthal. Joint State Theorems for Public-Key Encryption and Digitial Signature Functionalities with Local Computation. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF 2008)*. IEEE Comp. Soc. Press, 2008.
32. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proc. 2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166, Berlin (Germany), 1996. Springer-Verlag.
33. G. Lowe. Casper: A compiler for the analysis of security protocols. In *Proc. 10th Computer Security Foundations Workshop (CSFW'97)*. IEEE Comp. Soc. Press, 1997.
34. G. Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th Computer Security Foundations Workshop (CSFW'97)*, pages 18–30, Rockport (Massachusetts, USA), 1997. IEEE Computer Society Press.
35. A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, 1982.
36. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*, pages 166–175, 2001.
37. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communication of the ACM*, 21(12):993–999, 1978.
38. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299:451–475, 2003.
39. S. Schneider. Security properties and CSP. In *Proc. of the Symposium on Security and Privacy*, pages 174–187, Oakland, 1996. IEEE Computer Society Press.
40. H. Seidl and K. N. Verma. Flat and one-variable clauses: Complexity of verifying cryptographic protocols with single blind copying. In *Proc. 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'04)*, volume 3452 of *LNCS*. Springer, 2005.
41. D. X. Song. Athena: A new efficient automatic checker for security protocol analysis. In *Proc. 12th Computer Security Foundations Workshop (CSFW'99)*, Mordano, Italy, June 1999. IEEE Computer Society Press.

A Completeness of our simplification rules

Let $T_1 \subseteq T_2 \subseteq \ldots \subseteq T_n$. Given a left-minimal proof π of $T_i \vdash u$, we say that π is a proof of *level* j if $j = \min\{k \mid T_k \vdash u \text{ and } 1 \leq k \leq n\}$.

Lemma 9 *If $T_i \vdash u$ then there is a simple proof of it.*

Proof The notion of simple proof given in [23] is different from ours. However, a simple proof (according to the definition given in [23]) necessarily satisfies the two first conditions of our definition. Hence by using their result, we know that if $T_i \vdash u$ then there is a proof π of it which satisfies the conditions 1 and 2 of our definition. Now, let π be a proof of level j that satisfies the conditions 1 and 2 of Definition 16. We show that there exists a simple proof π' of $T_i \vdash u$ having the same level, i.e. j. We show this result by induction on m where m represents the number of nodes in π that violates condition 3.

Base case: $m = 0$. In such a case, we easily conclude. Indeed since $m = 0$, we have that π satisfies the condition 3. Thus, by definition, π is a simple proof.

Induction step: $m > 0$. In such a case, we show that we can transform the proof π into a proof π' having the same level and such that the number of nodes violating condition 3 is $m-1$. Then, it will be easy to conclude by applying the induction hypothesis. Let π_1 be a subproof of π whose root corresponds to the node that violates condition 3. We consider, among all these subproofs, one which is minimal in the sense that in π_1, the only node that violates the condition is its root.

The term $\langle u_1, u_2 \rangle$ that violates the condition is obtained by a decomposition rule whereas it is not immediately followed by a projection rule. This means that it is followed by a composition rule. We illustrate the situation when this last rule is an encryption rule. The proof π_1 has the following form:

$$\pi_1 := \left\{ \begin{array}{c} \dfrac{\dfrac{\cdots}{T_i \vdash \langle u_1, u_2 \rangle}\ \textsf{decompo.} \qquad T_i \vdash v}{T_i \vdash \{\langle u_1, u_2 \rangle\}_v}\ \textsf{compo.} \end{array} \right.$$

The idea is to replace this subproof π_1 of π by π_1' obtained by decomposing the term $\langle u_1, u_2 \rangle$ with the projection rules until we obtain terms not headed with the symbol $\langle\ \rangle$. Then, by using the pairing rule, we can build again the term $\langle u_1, u_2 \rangle$. Lastly, we apply the composition rule as in π_1. The proof π_1' obtained in this way has the same level that π_1. Hence, the proof π', obtained from π by replacing the subproof π_1 by π_1', is left-minimal. It is clear that condition 2 is satisfied since the composition rules introduced in π_1' are not directly followed by a decomposition rule. Lastly, we have removed one node violating condition 3 without introducing any such node. This allows us to conclude by using our induction hypothesis. $\square$

Let $\mathcal{C}$ be a constraint system and $x \in \mathit{vars}(\mathcal{C})$. We define T_x as follows

$$T_x = \min\{T \mid (T \Vdash u) \in \mathcal{C} \text{ and } x \in \mathit{vars}(u)\}.$$

Note that, by definition of a constraint system, T_x is well-defined.

Lemma 2 *Let $\mathcal{C}$ be an unsolved constraint system, θ be a solution of $\mathcal{C}$ and $T_i \Vdash u_i$ be a minimal unsolved constraint of $\mathcal{C}$. Let u be a term. If there is a simple proof of $T_i\theta \vdash u$ having the last rule an axiom or a decomposition then there is $t \in St(T_i) \smallsetminus \mathcal{X}$ such that $t\theta = u$.*

Proof Consider a simple proof π of $T_i\theta \vdash u$. Let j be minimal such that the proof π' obtained from π by replacing T_i with T_j is a proof of $T_j\theta \vdash u$. Depending on the last applied rule in the proof, we consider two cases.

– Either the last rule is an axiom.
  Then $u \in T_j\theta$ and hence there is $t \in T_j$ such that $t\theta = u$. If t is a variable then $T_t \Vdash t$ is a constraint in $\mathcal{C}$ with $T_t \subsetneq T_j$ (thanks to the definition of a constraint system). Hence $T_t\theta \vdash t\theta$, that is $T_t\theta \vdash u$, which contradicts the minimality of j.
– Or the last rule is a decomposition.
  Suppose w.l.o.g. that it is a symmetric decryption. Then, in such a case, there exists w such that $T_j\theta \vdash \mathrm{enc}(u,w)$ and $T_j\theta \vdash w$. By simplicity of the proof, the last rule applied to obtain $\mathrm{enc}(u,w)$ can not be a composition. Hence, it is either an axiom or a decomposition. Then, applying the induction hypothesis we have that there is $t \in St(T_j)$, t not a variable, such that $t\theta = \mathrm{enc}(u,w)$. It follows that $t = \mathrm{enc}(t',t'')$ with $t'\theta = u$. If t' is a variable then $T_{t'}\theta \vdash t'\theta$, that is $T_{t'}\theta \vdash u$ which contradicts the minimality of j. Hence t' is not a variable. For the other decomposition rules, the same reasoning holds. □

In order to prove Lemma 3, we need the following lemma.

Lemma 10 *Let T be a set of terms and u be a term such that $T \vdash u$. Then, we have that $plaintext(u) \subseteq plaintext(T)$.*

Proof let π be a proof of $T \vdash u$. We prove this result by induction on the depth of π. We can have:

– The last rule is an axiom. Then $u \in T$, thus $plaintext(u) \subseteq plaintext(T)$.
– The last rule is a composition. Suppose for example that it is the symmetric encryption rule. Then $u = \mathrm{enc}(u_1,u_2)$, $T \vdash u_1$ and $T \vdash u_2$. By definition, we have that $plaintext(u) = plaintext(u_1)$. Hence, we easily conclude by applying our induction hypothesis on $T \vdash u_1$. The other cases are similar.
– The last rule is a decomposition. Suppose for example that it is the symmetric decryption rule. In such a case, we have that $T \vdash \mathrm{enc}(u,v)$ and $T \vdash v$ for some term v. By induction hypothesis, $plaintext(\mathrm{enc}(u,v)) \subseteq plaintext(T)$. Hence, we easily conclude that $plaintext(u) \subseteq plaintext(T)$. The other cases are similar. □

Let t be a term, we denote by $\mathsf{comp}(t)$ the components of the term t. This notion is formally defined as follows: $\mathsf{comp}(\langle t_1,t_2\rangle) = \mathsf{comp}(t_1) \cup \mathsf{comp}(t_2)$ and $\mathsf{comp}(t) = t$ otherwise.

Lemma 3 *Let $\mathcal{C}$ be an unsolved constraint system, θ be a solution of $\mathcal{C}$ and $T_i \Vdash v_i$ be a minimal unsolved constraint of $\mathcal{C}$ such that*

1. *for all $t_1,t_2 \in St(T_i)$ such that $t_1 \neq t_2$*
$$t_1\theta = t_2\theta \ \textit{ implies } \ t_1 \textit{ or } t_2 \textit{ is a variable or a pair}$$
2. *for all $\mathrm{enca}(t_1,t_2) \in St(T_i)$ and $\mathrm{priv}(t_3) \in plaintext(T_i)$*
$$t_2\theta = t_3\theta \textit{ implies } t_2 = t_3.$$

Assume $u_i \in St(T_i) \smallsetminus \mathcal{X}$ and $T_i\theta \vdash u_i\theta$. Then $T_i \cup \{x \mid T \Vdash x \in \mathcal{C}, T \subsetneq T_i\} \vdash u_i$.

For any T_i left-hand side of a constraint system $\mathcal{C}$, we define
$$T_i^+ = T_i \cup \{x \mid T \Vdash x \in \mathcal{C}, T \subsetneq T_i\} \vdash u_i.$$

Proof Let j be minimal such that $T_j\theta \vdash u_i\theta$. Thus $j \leq i$ and $T_j \subseteq T_i$. Consider a simple proof of $T_j\theta \vdash u_i\theta$. We reason by induction on the depth of the proof. We can have that:

- The proof is reduced to an application of the rule axiom possibly followed by several application of the projection rules until the resulting term is not a pair. Since the proof is a simple proof, we have that $u_i\theta$ is not a pair. Hence, u_i is not a pair.
  There exists $t \in T_j$ such that $u_i\theta \in \mathsf{comp}(t\theta)$. Either $u_i\theta = t'\theta$ for some $t' \in \mathsf{comp}(t) \smallsetminus \mathcal{X}$ or $u_i\theta \in \mathsf{comp}(x\theta)$ for some $x \in \mathsf{comp}(t) \cap \mathcal{X}$. In the first case, we easily deduce that neither u_i nor t' is a pair or a variable and hence by hypothesis (point 1), we have that $u_i = t'$ and hence $T_i^+ \vdash u_i$. In the second case, we have that $T_x\theta \vdash x\theta$. Thus $T_x\theta \vdash u_i\theta$ which contradicts the minimality of j, since $T_x \subsetneq T_j$.
- The proof ends with an application of a decomposition rule possibly followed by several applications of the projection rules until the resulting term is not a pair. Note that, since the proof is a simple proof, we have that $u_i\theta$ is not a pair. Hence u_i is not a pair.

  Symmetric decryption rule. In such a case there exist w_1, w_2 such that $T_j\theta \vdash \mathrm{enc}(w_1, w_2)$, $T_j\theta \vdash w_2$ and $u_i\theta \in \mathsf{comp}(w_1)$. The last rule applied to obtain $T_j\theta \vdash \mathrm{enc}(w_1, w_2)$ was not a composition by simplicity of the proof. We can hence apply Lemma 2 and obtain that there is $t \in St(T_j) \smallsetminus \mathcal{X}$ such that $t\theta = \mathrm{enc}(w_1, w_2)$. Since t is not a variable, we have that $t = \mathrm{enc}(t_1, t_2)$ with $t_1\theta = w_1$ and $t_2\theta = w_2$. Either $u_i\theta = p\theta$ for some $p \in \mathsf{comp}(t_1) \smallsetminus \mathcal{X}$ or $u_i\theta \in \mathsf{comp}(x\theta)$ for some $x \in \mathsf{comp}(t_1) \cap \mathcal{X}$. In the second case, we have that $T_x\theta \vdash x\theta$. Thus $T_x\theta \vdash u_i\theta$ which contradicts the minimality of j, since $T_x \subsetneq T_j$. In the first case, we easily deduce that neither u_i nor p is a variable or a pair and hence by hypothesis (point 1), we have that $u_i = p$. We can apply the induction hypothesis on $T_j\theta \vdash \mathrm{enc}(t_1, t_2)\theta$ (this subproof is simple) to obtain that $T_i^+ \vdash \mathrm{enc}(t_1, t_2)$.
  Now, if t_2 is a variable then $t_2 \in T_i^+$, thus $T_i^+ \vdash t_2$. Otherwise, if t_2 is not a variable then, by induction hypothesis on $T_j\theta \vdash t_2\theta$ (this subproof is a simple one), we obtain $T_i^+ \vdash t_2$. Hence, in both cases, we obtain that $T_i^+ \vdash t_2$. Then, together with $T_i^+ \vdash \mathrm{enc}(t_1, t_2)$ and $u_i \in \mathsf{comp}(t_1)$, it follows that $T_i^+ \vdash u_i$.

  Signature rule. This case is similar to the previous one.

  Asymmetric decryption rule. In such a case there exist w_1, w_2 such that $T_j\theta \vdash \mathrm{enca}(w_1, w_2)$, $T_j\theta \vdash \mathrm{priv}(w_2)$ and $u_i\theta \in \mathsf{comp}(w_1)$. We easily obtain (as in the previous case) that $T_i^+ \vdash \mathrm{enca}(t_1, t_2)$ and $T_i^+ \vdash \mathrm{priv}(t_3)$ with $t_2\theta = t_3\theta$. Thanks to Lemma 10, we deduce that $\mathrm{priv}(t_3) \in \mathit{plaintext}(T_i)$. Then by hypothesis (point 2), we deduce that $t_2 = t_3$, it follows that $T_i^+ \vdash t_1$ and since $u_i \in \mathsf{comp}(t_1)$, we have that $T_i^+ \vdash u_i$.
- The last rule is a composition.
  Suppose that it is the symmetric encryption rule. Then $u_i\theta = \mathrm{enc}(w_1, w_2)$ and $T_j\theta \vdash w_1$ and $T_j\theta \vdash w_2$. Since u_i is not a variable, we have that $u_i = \mathrm{enc}(v_1', v_2')$, $v_1'\theta = w_1$ and $v_2'\theta = w_2$. If v_1' (resp. v_2') is a variable then v_1' (resp. v_2') is in T_i^+ (this is because $v_j \in St(T_i)$). Otherwise, we apply our induction hypothesis (note that the two subproofs are simple). Hence, in both cases, we have that $T_i^+ \vdash v_1'$ and also that $T_i^+ \vdash v_2'$. Hence, we easily deduce that $T_i^+ \vdash u_i$. For the other composition rules the same reasoning holds. $\square$

B Proofs of our Composition Result

B.1 Existence of a solution without any mixing

Before proving Lemma 5, we first state some useful lemmas. Lemma 11 can be proved by induction on the algorithm that computes the most general unifier (see [35]).

Lemma 11 *Let T_1 and T_2 be two sets of terms having disjoint encryption and consider $t, t' \in EncSt(T_1 \cup T_2)$ two terms that are unifiable. Let $\sigma = \mathrm{mgu}(t, t')$. Then $T_1\sigma$ and $T_2\sigma$ have disjoint encryption and $names(T_i\sigma) \subseteq names(T_i)$ for $i = 1, 2$.*

Lemma 12 *Let KC be a set of extended names, $\mathcal{C}$ be a constraint system satisfying the plaintext origination property such that $\mathsf{KC} \cap plaintext(\mathsf{lhs}(\mathcal{C})) = \emptyset$ and σ be a substitution. If $\mathcal{C}\sigma$ is satisfiable, then $\mathsf{KC} \cap plaintext(\mathsf{lhs}(\mathcal{C}\sigma)) = \emptyset$.*

Proof Suppose $\mathcal{C}\sigma$ is satisfiable. Let θ be a solution of $\mathcal{C}\sigma$ and let $\theta' = \sigma\theta$. We show the result by contradiction. Assume that there exists a constraint $T \Vdash u \in \mathcal{C}$ such that $\mathsf{KC} \cap plaintext(T\sigma) \neq \emptyset$. This implies that $\mathsf{KC} \cap plaintext(T\sigma\theta) \neq \emptyset$, thus there exists $k \in \mathsf{KC}$ such that:

- either $k \in plaintext(T)$;
- or $k \in plaintext(x\theta')$ for some $x \in plaintext(T)$.

The first case is impossible by hypothesis. Let x be the minimal variable verifying such a condition, that is the variable that is introduced in plaintext by the minimal constraint. Let $T' \Vdash u' \in \mathcal{C}$ be the minimal constraint such that $x \in plaintext(u')$. We have that $T'\theta' \vdash u'\theta'$ since θ' is a solution of $\mathcal{C}$. We have that $k \in plaintext(u'\theta')$, thus by Lemma 10, we have that $k \in plaintext(T'\theta')$. Since $k \notin plaintext(T')$, this means that there exists $y \in plaintext(T')$ (note that y is smaller than x) such that $k \in plaintext(y\theta')$, contradiction. □

Lemma 13 *Let $\mathcal{C}$ be a constraint system satisfying the plaintext origination property. Let σ be a substitution. Then $\mathcal{C}\sigma$ satisfies the plaintext origination property.*

Proof Let $\mathcal{C} = T_1 \Vdash u_i, \ldots, T_n \Vdash u_n$ and σ be a substitution. Let $1 \leq i \leq n$ and x a variable such that $x \in plaintext(T_i\sigma)$. Note that since T_1 is necessarily ground (by definition of a constraint system, Definition 14), we have that $i > 1$. We have to show that there exists $j < i$ such that $x \in plaintext(u_j\sigma)$. We distinguish two cases:

- Either x is not introduced by σ, i.e. $x \notin \{vars(y\sigma) \mid y \in \mathrm{dom}(\sigma)\}$. In such a case, since $\mathcal{C}$ satisfies the plaintext origination property, we know that there exists $j < i$ such that $x \in plaintext(u_j)$. Thus, we have that $x \in plaintext(u_j\sigma)$.
- Otherwise x is introduced by σ, i.e. $x \in vars(y\sigma)$ for some $y \in \mathrm{dom}(\sigma)$. Moreover, since x occurs at a plaintext position, we have that $x \in plaintext(y\sigma)$ and $y \in plaintext(T_i)$. Since $\mathcal{C}$ satisfies the plaintext origination property, we have that there exists $j < i$ such that $y \in plaintext(u_j)$. From this, we easily conclude that $x \in plaintext(u_j\sigma)$. □

Lemma 5 *Let T_1 and T_2 be two sets of terms having disjoint encryption and* KC *be a set of extended names. Let $\mathcal{C}$ be a well-formed constraint system w.r.t. $St(T_1) \cup St(T_2)$ and* KC*. Let $\mathcal{C}'$ and σ be such that $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$ with $\mathcal{C}'$ satisfiable. Then, we have that*

1. *$T_1\sigma$ and $T_2\sigma$ have disjoint encryption,*
2. *$names(T_i\sigma) \subseteq names(T_i)$ for $i = 1, 2$, and*
3. *the constraint system $\mathcal{C}'$ is well-formed w.r.t. $St(T_1\sigma) \cup St(T_2\sigma)$ and* KC.

Proof We prove this result by case analysis on the simplification rule involved in the reduction $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$.

Case of the rule R_1*:* $\mathcal{C} = \mathcal{C}' \wedge T \Vdash u \;\rightsquigarrow\; \mathcal{C}'$. In such a case, we have that σ is the identity. Thus, the two first requirements are satisfied by hypothesis. Moreover, we have that

$$\mathsf{lhs}(\mathcal{C}') \cup \mathsf{rhs}(\mathcal{C}') \;\subseteq\; \mathsf{lhs}(\mathcal{C}) \cup \mathsf{rhs}(\mathcal{C}) \;\subseteq\; St(T_1) \cup St(T_2).$$

We have that $\mathsf{lhs}(\mathcal{C}') \subseteq \mathsf{lhs}(\mathcal{C})$ and by hypothesis we have also that $\mathsf{KC} \cap plaintext(\mathsf{lhs}(\mathcal{C})) = \emptyset$. Hence, we deduce that $\mathsf{KC} \cap plaintext(\mathsf{lhs}(\mathcal{C}')) = \emptyset$. It remains to establish the fact that $\mathcal{C}'$ satisfies the plaintext origination property. Let $T' \Vdash u' \in \mathcal{C}'$ and $x \in plaintext(T) \cap \mathcal{X}$. Let $T'' \Vdash u''$ be the minimal constraint of $\mathcal{C}$ (w.r.t. inclusion of the left-hand side) such that $x \in plaintext(u'')$ (note that $x \notin plaintext(T'')$). Either $T'' \Vdash u'' \in \mathcal{C}'$ and we easily conclude. Otherwise, we have that $T'' = T$ and $u'' = u$. By hypothesis, we know that $T \cup \{x \mid T' \Vdash x \in \mathcal{C}$ and $T' \subsetneq T\} \vdash u$. Thus, by Lemma 10, we have $plaintext(u) \subseteq plaintext(T) \cup \{x \mid T' \Vdash x \in \mathcal{C}$ and $T' \subsetneq T\}$. Since $x \notin plaintext(T)$, we have $x \in \{x \mid T' \Vdash x \in \mathcal{C}$ and $T' \subsetneq T\}$, which contradicts the minimality of $T'' \Vdash u''$ and allows us to conclude.

Case of the rule R_2 *or* R_3*:* Thanks to Lemma 11, the two first requirements are satisfied. We have that

$$\begin{array}{rcl}\mathsf{lhs}(\mathcal{C}') \cup \mathsf{rhs}(\mathcal{C}') & = & \{t\sigma \mid t \in \mathsf{lhs}(\mathcal{C}) \cup \mathsf{rhs}(\mathcal{C})\} \\ & \subseteq & \{t\sigma \mid t \in St(T_1) \cup St(T_2)\} \\ & \subseteq & St(T_1\sigma) \cup St(T_2\sigma)\end{array}$$

The plaintext origination condition is stable by application of a substitution thanks to Lemma 13. Thus, $\mathcal{C}\sigma$ satisfies this condition. Lastly, we have that $\mathcal{C}' = \mathcal{C}\sigma$ is satisfiable. Thus, thanks to Lemma 12, we easily deduce that $\mathsf{KC} \cap plaintext(\mathsf{lhs}(\mathcal{C}')) = \emptyset$.

Case of the rule R_4*:* $\mathcal{C} = \mathcal{C}_1 \wedge T \vdash u \rightsquigarrow_\sigma \mathcal{C}_1\sigma \wedge T\sigma \Vdash u\sigma = \mathcal{C}'$ where $\sigma = \mathrm{mgu}(t_2, t_3)$, $\mathrm{enca}(t_1, t_2) \in St(T)$, $\mathrm{priv}(t_3) \in plaintext(T)$ and $t_2 \neq t_3$. In such a case, we have that $t_3 = a$ for some agent name a and thus t_2 is a variable, say y. Hence, we have that $\sigma = \{y \mapsto a\}$. Clearly, $T_1\sigma$ and $T_2\sigma$ have disjoint encryption and $names(T_i\sigma) \subseteq names(T_i)$ since $a \in \mathcal{A}$ and $\mathcal{A} \cap \mathcal{N} = \emptyset$. It remains to show that $\mathcal{C}'$ is well-formed w.r.t. $St(T_1\sigma) \cup St(T_2\sigma)$. The two first points are easy to check and the last one is a direct consequence of Lemma 12.

Case of the rule R_5*:* This rule leads to a constraint system $\mathcal{C}'$ that is not satisfiable.

Case of the rule R_6*:* In such a case, σ is the identity, thus the two first requirements are satisfied. Clearly, $\mathsf{lhs}(\mathcal{C}') \cup \mathsf{rhs}(\mathcal{C}') \subseteq St(T_1) \cup St(T_2)$. Since we have that $plaintext(\mathsf{rhs}(\mathcal{C}')) \supseteq plaintext(\mathsf{rhs}(\mathcal{C}))$, the plaintext origination is satisfied. Lastly, since $\mathsf{lhs}(\mathcal{C}') = \mathsf{lhs}(\mathcal{C})$, we easily deduce that $\mathsf{KC} \cap plaintext(\mathsf{lhs}(\mathcal{C}')) = \emptyset$. ☐

Lemma 6 *Let $\mathcal{C}$ be a constraint system in solved form and* DEq *be a finite set of disequations such that τ is a solution of $\mathcal{C} \wedge \mathsf{DEq}$. There exists a solution τ' of $\mathcal{C} \wedge \mathsf{DEq}$ such that for every variable $x \in \mathrm{dom}(\tau')$, we have that $x\tau' \in \mathsf{Sinit}$.*

We define a transformation function $\overline{\cdot}$ that simplifies conjunction of disequations as follows:

$$\begin{aligned}
\overline{\phi \wedge [f(m_1, \ldots, m_k) \neq g(m'_1, \ldots, m'_l)]} &= \mathsf{true} \quad \text{if } f \neq g\\
\overline{\phi \wedge [f(m_1, \ldots, m_k) \neq f(m'_1, \ldots, m'_k)]} &= \overline{\phi} \wedge (\overline{[m_1 \neq m'_1]} \vee \cdots \vee \overline{[m_k \neq m'_k]})\\
\overline{\phi \wedge [x \neq m]} &= \begin{cases} \neg\,\mathsf{true} & \text{if } m = x\\ \overline{\phi} & \text{if } x \in vars(m), x \neq m\\ \overline{\phi} \wedge [x \neq m] & \text{otherwise} \end{cases}
\end{aligned}$$

We obtain a formula $\overline{\phi}$ of the form true, $\neg\,\mathsf{true}$ or

$$\begin{array}{ll}
 & ([x_{1,1} \neq m_{1,1}] \vee \cdots \vee [x_{1,k_1} \neq m_{1,k_1}])\\
\wedge & ([x_{2,1} \neq m_{2,1}] \vee \cdots \vee [x_{2,k_2} \neq m_{2,k_2}])\\
\vdots & \qquad\qquad \vdots\\
\wedge & ([x_{l,1} \neq m_{l,1}] \vee \cdots \vee [x_{l,k_l} \neq m_{l,k_l}])
\end{array}$$

where $x_{i,j} \notin vars(m_{i,j})$.

Now, we are able to establish Lemma 6.

Proof Let T_1 the smallest left-hand side of $\mathcal{C}$. Note that, since $\mathsf{init} \in T_1$, we have that $T_1 \vdash t$ for any $t \in \mathsf{Sinit}$ and the set Sinit is infinite. Moreover, for any substitution σ such that $x\sigma \in \mathsf{Sinit}$ for every variable $x \in \mathrm{dom}(\sigma)$, we have that σ is a solution of $\mathcal{C}$ since $\mathcal{C}$ is in solved form.

We show that any formula of the form $\phi = [x_1 \neq m_k] \vee \cdots \vee [x_n \neq m_k]$ such that $x_i \notin vars(m_i)$ has a solution σ such that $x\sigma \in \mathsf{Sinit}$ for every variable $x \in vars(\phi)$. This is done by induction on the number of variables in ϕ. Note that this allows us to conclude the proof of Lemma 6.

Base case. If ϕ has exactly one variable, then $\phi = [x \neq m_1] \wedge \cdots \wedge [x \neq m_k]$ with $x \notin vars(m_i)$. Thus all m_i are ground terms. Consider a term $m \in \mathsf{Sinit}$ such that $m \neq m_i$ for $1 \leq i \leq k$. We have $T_1 \vdash m$. The substitution τ' such that $y\tau' = m$ for any $y \in vars(\phi)$ is a solution of ϕ.

Inductive case. $\phi = [x \neq m_1] \wedge \cdots \wedge [x \neq m_k] \wedge [x \neq t_1] \wedge \cdots \wedge [x \neq t_l] \wedge \phi'$ where

- the m_i are ground,
- $x \notin vars(t_i)$ and $vars(t_i)$ is non empty,
- ϕ' is of the form $[x_1 \neq u_1] \wedge \cdots \wedge [x_s \neq u_s]$ with $x_i \notin vars(u_i)$ and $x \neq x_i$.

Consider $m \in \mathsf{Sinit}$ such that $m \neq m_i$ for $1 \leq i \leq k$. We have $T_1 \vdash m$. Let $\sigma = \{^m/_x\}$. We consider $\phi\sigma$.

- Each formula $[x \neq m_i]\sigma$ is true
- Let $\phi'' = \phi'\sigma$. Note that ϕ'' is of the right form, that is ϕ'' is a conjunction of formulas of the form $[y \neq u]$ with $y \notin vars(u)$.
- Let $I = \emptyset$. For each $1 \leq i \leq l$, we consider the formula $m \neq t_i\sigma$. There is a variable $y_i \in vars(t_i\sigma)$. We choose one occurrence p_i of y_i in $t_i\sigma$, that is $t_i\sigma|_{p_i} = y_i$. If p_i is not a path in m then the formula $[m \neq t_i\sigma]$ is always true. Otherwise, we define $m'_i = m|_{p_i}$ and we let $I := I \cup \{i\}$.

We consider the formula $\psi = \phi'' \wedge \bigwedge_{i \in I} [y_i \neq m'_i]$. We have that $y_i \notin \mathit{vars}(m'_i)$ since m'_i is ground. Moreover, ψ does not contain the variable x thus ψ has strictly less variables than ϕ. We deduce by induction hypothesis that there is a solution θ to ψ such that $x\theta \in \mathsf{Sinit}$ for any $x \in \mathit{vars}(\psi)$. Thus, we have that $\sigma\theta$ is a solution of the right form to ϕ, which concludes the proof. $\square$

B.2 Getting rid of the terms coming from Π_2

Lemma 7 (locality) *Let T be a set of terms and u be a term such that $T \vdash u$. Let π be a proof of $T \vdash u$ which is minimal w.r.t. its number of nodes. Then π only involves terms in $\mathit{St}(T \cup \{u\})$. Moreover, if π ends with a decomposition rule or the axiom rule then π only involves terms in $\mathit{St}(T)$ and $u \in \mathit{St}_{plain}(T)$.*

Proof Let π be a proof of $T \vdash u$ which is minimal *w.r.t.* to its number of nodes. We show the result by induction on π. We can have that:

- The last rule is an axiom. In such a case, we easily conclude.
- The last rule is a composition. Suppose for example that it is the symmetric encryption rule. In such a case, we have that $u = \mathrm{enc}(u_1, u_2)$. Let π_1 (resp. π_2) be the subproof of π ending on $T \vdash u_1$ (resp. $T \vdash u_2$). By induction hypothesis, we know that π_1 (resp. π_2) only involves terms in $\mathit{St}(T \cup \{u_1\})$ (resp. $\mathit{St}(T \cup \{u_2\})$). Hence, we easily deduce that π only involves terms in $\mathit{St}(T \cup \{u\})$. The same reasoning holds for the other composition rules.
- The last rule is a decomposition. Suppose for example that it is the symmetric decryption rule. In such a case, we have that

$$\frac{\pi_1 = \left\{ \dfrac{\cdots}{T \vdash \mathrm{enc}(u, v)} \right. \qquad \pi_2 = \left\{ \dfrac{\cdots}{T \vdash v} \right.}{T \vdash u}$$

  Note that, by minimality of π, the proof π_1 necessarily ends with a decomposition rule. Hence, by induction hypothesis, we know that π_1 only involves terms in $\mathit{St}(T)$ and also that $\mathrm{enc}(u, v) \in \mathit{St}_{plain}(T)$. In particular, we have $v \in \mathit{St}(T)$. By induction hypothesis, we know that π_2 only involves terms of $\mathit{St}(T \cup \{v\})$ thus terms of $\mathit{St}(T)$. Thus we easily deduce that π only involves terms of $\mathit{St}(T)$ and also that $u \in \mathit{St}_{plain}(T)$. For the other decomposition rules a similar reasoning holds. In the case of the asymmetric decryption rule, we have that $v \in \mathit{St}(T)$ since, by induction hypothesis, a term of the form $\mathrm{priv}(v')$ can only be obtained by the axiom rule or a decomposition rule. $\square$

Lemma 8 *Let T_0 be a set of terms such that $\mathit{names}(T_0) \cap \mathsf{Names} = \emptyset$ and $\mathsf{init} \in T_0$. Let v be a term such that $\mathit{plaintext}(v) \subseteq T_0 \cup \mathsf{Names}$ and $\mathit{EncSt}(v) \subseteq \mathit{EncSt}(\mathsf{ETerms})$. Then, we have that $T_0 \vdash \overline{v}$.*

The proof below relies on the notion of component which is formally defined in Appendix A.

Proof We show that for every $p \in \mathsf{comp}(v)$, we have that $T_0 \vdash \overline{p}$. By definition of $\overline{\cdot}$, we have that

$$\{\overline{p} \mid p \in \mathsf{comp}(v)\} = \{p' \mid p' \in \mathsf{comp}(\overline{v})\}.$$

Then, we can easily deduce that $T_0 \vdash p'$ for every $p' \in \mathsf{comp}(\overline{v})$, and thus $T_0 \vdash \overline{v}$.

Let $p \in \mathsf{comp}(v)$. We distinguish three cases:

1. p is of the form $\mathrm{enc}(w_1, w_2)$, $\mathrm{enca}(w_1, w_2)$ or $\mathrm{sign}(w_1, w_2)$. In such a case, since $\mathit{EncSt}(v) \subseteq \mathit{EncSt}(\mathsf{ETerms})$, we have that $\overline{p} = \mathsf{init}$, thus $T_0 \vdash \overline{p}$.
2. p is of the form a (or $\mathrm{priv}(a)$) with $a \in \mathcal{A}$. In such a case, we have that $p \in T_0$ and $\overline{p} = p \in T_0$.
3. p is a name, i.e. $p \in \mathcal{N}$. We have that $p \in \mathit{plaintext}(v)$ and $p \in T_0 \cup \mathsf{Names}$, thus $T_0 \vdash \overline{p}$. This allows us to conclude. □