

Ștefan Ciobâcă, Véronique Cortier

Protocol composition for arbitrary primitives

Research Report LSV-10-09

April, 2010

Laboratoire
Spécification
et
Vérification



**CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE**

**Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France**

Protocol composition for arbitrary primitives

Ștefan Ciobâcă
 LSV, ENS Cachan & CNRS
 Email: stefan.ciobaca@gmail.com

Véronique Cortier
 LORIA, CNRS & INRIA
 Email: cortier@loria.fr

Abstract—We study the composition of security protocols when protocols share secrets such as keys.

We show (in a Dolev-Yao model) that if two protocols use disjoint cryptographic primitives, their composition is secure if the individual protocols are secure, even if they share data. Our result holds for any cryptographic primitives that can be modeled using equational theories, such as encryption, signature, MAC, exclusive-or, and Diffie-Hellman.

Our main result transforms any attack trace of the combined protocol into an attack trace of one of the individual protocols. This allows various ways of combining protocols such as sequentially or in parallel, possibly with inner replications.

As an application, we show that a protocol using pre-established keys may use any (secure) key-exchange protocol without jeopardizing its security, provided that they do not use the same primitives. This allows us, for example, to securely compose a Diffie-Hellman key exchange protocol with any other protocol using the exchanged key, provided that the second protocol does not use the Diffie-Hellman primitives.

We also explore tagging, which is a way of forcing the disjointness of two protocols which share cryptographic primitives such as encryption. We explain why composing protocols which use tagged cryptographic primitives like encryption and hash functions is secure by reducing this problem to the previous one.

I. INTRODUCTION

Security protocols aim at ensuring security properties (for example confidentiality or authentication) of communications over public networks. Their design is error-prone due to the fact that they are used in arbitrary environments, with possibly malicious behaviors. It is known for example that a small variation in the design of a protocol may open the way to an attack (see e.g. [1]).

Formal methods have demonstrated their usefulness when designing and analyzing security protocols. They indeed provide rigorous frameworks and techniques that have led to the discovery of new flaws [1], [2], [3] and to careful security analysis (see e.g. [4], [5]). While insecurity is undecidable in general [6], several decision procedures (sometimes incomplete) have been proposed for automatically analyzing the security of protocols. For example, checking secrecy and authentication-like properties has been shown to be NP-complete [7] for a bounded number of sessions. Blanchet has developed a procedure based on clause resolution [8] for analyzing protocols for an unbounded number of sessions. Several tools [9], [2], [10] have been developed and successfully applied to checking the security of protocols.

However, most of existing techniques are dedicated to the analysis of a single protocol, without taking into account other protocols which may be used at the same time.

This is unrealistic for several reasons. Firstly, a number of protocols are verified under the assumption that agents share some pre-distributed keys (e.g. public keys or symmetric keys between agents and servers). But these keys might have been established by some other sub-protocols. There is no guarantee that a protocol remains secure if a specific key-exchange protocol is used to establish the keys, even if both protocols have been proven secure in isolation.

Secondly, even apparently isolated protocols might interact in unexpected ways. For example, a user might choose the same password for two different network services, or a server might use the same key for different protocols. Even if the network services (or the different protocols) were proven secure in isolation, there is no security guarantee which carries over when they share keys or passwords.

Furthermore, even assuming that we can produce a global model of all protocols which are used in a certain setting, it might be unrealistic to formally verify such a collection of protocols in its entirety due to computational constraints.

Therefore more modular reasoning about security is desirable, where we can infer security guarantees for the composition of protocols from the security guarantees of the individual protocols.

The goal of our paper is to study the composition of protocols. We use *composition* to refer to arbitrary ways of interleaving protocols, in particular in parallel or sequentially. For example, given a protocol P_1 that has been proven secure assuming pre-established keys or assuming some secure channel, we wish to study under which conditions P_1 remains secure if it uses P_2 as a sub-protocol to establish some of keys.

Related work. There are a number of papers studying the secure composition of security protocols in a symbolic, Dolev-Yao model [11], [12], [13], [14], [15], [16] and in the computational model [17], [18]. We explain how our result compares to the existing work in Section VI.

Our contributions. We propose a generic composition result for arbitrary cryptographic primitives which can be modeled by equational theories. More precisely, we show that an attack trace against the composition of two protocols can be

transformed into an attack trace on one of the two protocols. For the clarity of the exposition we concentrate on secrecy properties although we believe that our result carries over to other trace properties such as authentication.

Our main theorem is generic in the sense that the composition can be any interleaving of actions from the two protocols: for example, the composition can be parallel or sequential, possibly with nested replication. In particular, we capture the case where a protocol uses a sub-protocol to e.g. establish keys.

The composition theorem holds for any cryptographic primitives which can be modeled by equational theories, provided that the signatures of the two composed protocols are disjoint. This allows us to handle many cryptographic primitives such as symmetric and asymmetric encryption, hash functions, messages authentication codes, signatures, blind signatures, re-encryption, zero-knowledge proofs and others [19], [20]. We can also allow some common primitives between the two protocols, such as encryption and hash, provided that they are tagged.

As a consequence, we can for example easily compose a protocol using Diffie-Hellman exponentiation for establishing symmetric keys, together with any protocol making use of pre-established keys.

Applications. Our main composition result can be used in different contexts. As an application, we study the case of key-exchange protocols. We first consider the case where a key-exchange protocol is used to establish shared long-term keys. Assume that $P = \nu n \cdot (P_1 \mid P_2)$ is a protocol that establishes a key between two participants. P_1 intuitively denotes the first participant, P_2 denotes the second participants, \mid denotes the fact that P_1 and P_2 run in parallel, \cdot denote sequential composition and νn means that P_1 and P_2 share some secret n . The role of P is to establish a shared key between P_1 and P_2 . Assume that the key will be stored in the variable y_1 for P_1 and in the variable y_2 for P_2 .

An important question is the following one: which properties should be satisfied by P in order to be safely used within any other protocol? As expected, we retrieve the fact that the established key (stored in y_1 for P_1 and in y_2 for P_2) should remain secret to an attacker, but we also point out two other important properties which are not always checked in security proofs of the literature.

We show that whenever P satisfies our identified properties and whenever a protocol Q is secure assuming pre-established keys (e.g. if Q preserves the secrecy of some data s):

$$Q = \nu k \cdot ((y_1 := k) \cdot Q_1 \mid (y_2 := k) \cdot Q_2) \models \text{Secret}(s)$$

then Q remains secure when running P as subprotocol for establishing the secret key (in y_1 for the first participant and in y_2 for the second participant):

$$\nu n \cdot (P_1 \cdot Q_1 \mid P_2 \cdot Q_2) \models \text{Secret}(s).$$

We also consider the case where a key-exchange protocol is used within each session for establishing a secure channel. We show that if a protocol Q' is secure assuming a secure channel:

$$Q' \models (\nu k \cdot ((y_1 := k) \cdot Q_1 \mid (y_2 := k) \cdot Q_2)) \models \text{Secret}(s)$$

then Q' remains secure when running P as subprotocol:

$$!(\nu n \cdot (P_1 \cdot Q_1 \mid P_2 \cdot Q_2)) \models \text{Secret}(s).$$

We describe our setting in Section II. We state our generic composition theorem in Section III, providing counter-examples when protocols are not carefully composed. In Section IV, we illustrate our main theorem with the case of key-exchange protocols. We explain how to compose protocols with common tagged primitives in Section V. We discuss related work in Section VI.

II. MODEL

We first introduce a process algebra for security properties. The process algebra closest to ours is the applied π -calculus [21]. However, the applied π -calculus is not adequate in our case since it makes formulating our main theorem unnecessarily cumbersome. The main differences between our calculus and the applied π -calculus are the following ones:

- we add a *synchronization phase*, so that we can write $P \cdot Q$ for arbitrary processes P and Q . This is important to express the fact that a protocol first runs P before continuing with Q ,
- we consider only one public channel,
- only positive tests are allowed (no else branches).

A. Terms and deduction

The process algebra is parametrized by a signature \mathcal{F} which associates to each function symbol f its arity $ar(f)$. We assume that the signature contains at least a (regular) constant function symbol and that it contains a countably infinite set \mathcal{N} of special constant symbols, which we call *names* and which are used to represent data freshly generated during protocol executions. We also assume a countably infinite set of variables $\mathcal{X}_{all} = \mathcal{X} \cup \mathcal{X}_w$, disjoint from \mathcal{F} and such that \mathcal{X} and \mathcal{X}_w are disjoint countably infinite sets. Intuitively, the variables of \mathcal{X} will be used to describe the variables instantiated by the protocols while the variables of \mathcal{X}_w will be used to store the messages sent on the network.

The set of terms over a signature \mathcal{F} and over a set of variables \mathcal{X} is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and is defined as follows:

t, t_1, \dots	$::=$	terms	
	\mid	x	variable $x \in \mathcal{X}$
	\mid	$f(t_1, \dots, t_k)$	application of symbol $f \in \mathcal{F}, ar(f) = k$

A term is *ground* if it contains no variable. As usual, we denote by $\{x_1 \mapsto t_1, \dots, x_k \mapsto t_k\}$ the substitution σ that replaces the variable x_i with the term t_i . The *domain* of σ , denoted by $dom(\sigma)$ is the set $\{x_1, \dots, x_k\}$. The substitution σ is ground if each t_i is ground. We let $\mathbf{E} = \{l_i = r_i\}_{i \in \{1, \dots, n\}}$ be an equational theory, where $l_i, r_i \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}, \mathcal{X})$ ($1 \leq i \leq n$) are not allowed to contain names. We say that two terms s and t are equal in the equational theory \mathbf{E} and we write $s =_{\mathbf{E}} t$ if $s = t$ is a consequence of \mathbf{E} in the first order theory of equality. We denote by $[t]_{=\mathbf{E}}$ the equivalence class

of a term t modulo E . Note that such an equational theory is stable by replacement of names by arbitrary terms.

We write $u \in_E S$ if there exists $t \in S$ such that $u \equiv_E t$. Given a finite set S , its cardinality is denoted by $|S|$.

Processes are executed within an environment formed of a *frame* φ that contains messages sent over the network (as in the applied π -calculus) and a *binding substitution* σ whose domain is a subset of the free variables of the process.

In what follows, we always assume a *frame* φ to be a ground substitution whose domain is included in \mathcal{X}_w . An equational theory typically describes the properties of the primitives and defines what an attacker can *deduce* from a set of messages, represented by a frame.

Definition 1 (deduction): Let φ be a frame. We say that $t \in \mathcal{T}(\mathcal{F})$ is deducible from φ with recipe $r \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}, \text{dom}(\varphi))$ in the equational theory E and we write $\varphi \vdash_E^r t$ if $r\varphi \equiv_E t$. If E is clear from context, we write only \vdash^r instead of \vdash_E^r . If we are not interested in the exact value of the recipe r , we also write \vdash_E or \vdash (if the equational theory is clear from context).

Note that we assume that during the deduction phase the intruder does not have access to the infinite set of names. This is not a restriction in our case, since we will only allow positive tests in processes; so the intruder can simply use any term in place of the names.

Example 1: Let $\mathcal{F}_{\text{DH}} = \{f, g, \text{mac}, n_I\} \cup \mathcal{N}$ be a signature where f and g are of arity 1 while mac is of arity 2 and n_I is of arity 0 (n_I represents a public data). Together with the equational theory

$$E_{\text{DH}} = \{f(g(y), x) = f(g(x), y)\}$$

the function symbols f and g model the Diffie-Hellman primitives ($f(x, y) = x^y \bmod p$, $g(y) = \alpha^y \bmod p$ for a generator α) while mac denotes a keyed hash function.

Let $\varphi_1 = \{w_1 \mapsto g(a), w_2 \mapsto g(b), w_3 \mapsto c\}$ where a, b, c are names. Then $\varphi_1 \vdash_{E_{\text{DH}}}^{f(w_1, w_3)} f(g(c), a)$ but $\varphi_1 \not\vdash_{E_{\text{DH}}} f(g(a), b)$.

Example 2: A classical example is the modeling of symmetric encryption. Let $\mathcal{F}_{\text{enc}} = \{\text{dec}, \text{enc}, m_I\} \cup \mathcal{N}$ be a signature where dec and enc are of arity 2 and represent respectively the decryption and encryption operator. m_I is of arity 0 and represents some public data. As usual, we may write $\{m\}_k$ instead of $\text{enc}(m, k)$.

The standard property of symmetric encryption/decryption is represented by the equational theory $E_{\text{enc}} = \{\text{dec}(\text{enc}(x, y), y) = x\}$. Let $\varphi_2 = \{w_1 \mapsto \{k_1\}_{k_2}, w_2 \mapsto k_2, w_3 \mapsto \{k_3\}_{k_1}\}$, where k_1, k_2, k_3 are names. Then $\varphi_2 \vdash_{E_{\text{enc}}}^{\text{dec}(w_3, \text{dec}(w_1, w_2))} k_3$.

B. Combination of equational theories

To prove our composition result for security protocols, we make use of some notions and results in term rewriting for disjoint equational theories. We recall here these notions and results.

Let \mathcal{F}_a and \mathcal{F}_b be two disjoint signatures and let E_a and E_b be two nontrivial equational theories over \mathcal{F}_a and respectively

\mathcal{F}_b . Let $E = E_a \cup E_b$. If $c \in \{a, b\}$, by \bar{c} we will denote the only element of the singleton set $\{a, b\} \setminus \{c\}$.

Definition 2 (pure term, pure context): We say that a term $t \in \mathcal{T}(\mathcal{F}_a \cup \mathcal{F}_b)$ is a pure (\mathcal{F}_c -)term if $t \in \mathcal{T}(\mathcal{F}_c)$ for some $c \in \{a, b\}$. Similarly, a context $C \in \mathcal{T}(\mathcal{F}_a \cup \mathcal{F}_b, \mathcal{X})$ is a pure (\mathcal{F}_c -)context if $C \in \mathcal{T}(\mathcal{F}_c, \mathcal{X})$ for some $c \in \{a, b\}$.

Definition 3 (alien subterms): Let $c \in \{a, b\}$ be such that $\text{root}(C) \in \mathcal{F}_c$. If $t = C[s_1, \dots, s_n]$ where C is a pure \mathcal{F}_c -context and $\text{root}(s_j) \in \mathcal{F}_{\bar{c}}$, we write $t = C[[s_1, \dots, s_n]]$ and we say that s_1, \dots, s_n are the alien subterms of t . Note that C and s_1, \dots, s_n are uniquely determined and that C cannot be the empty context (at least the root symbol of t is in C).

We define a function ‘collapse’ which associates to any term t a *collapsed* term s such that $s \equiv_E t$. The collapsed version of a term serves as a kind of “normal form” of the term.

Definition 4 (collapse): If $t = C[[t_1, \dots, t_n]]$ and $C[\text{collapse}(t_1), \dots, \text{collapse}(t_n)] = D[[s_1, \dots, s_k]]$, then $\text{collapse}(t)$ is defined recursively as follows:

1) if

$$D[n_{[s_1]=E}, \dots, n_{[s_k]=E}] \equiv_E n_{[s_j]=E}$$

for some j , then

$$\text{collapse}(t) = s_j$$

If there are several values j which satisfy the condition, we choose the minimal such j . The choice of j does not influence our results, but it makes the function *collapse* determined, which eases the proofs.

2) otherwise, if there exists no such j , we define

$$\text{collapse}(t) = C[\text{collapse}(t_1), \dots, \text{collapse}(t_n)]$$

If $t = \text{collapse}(t)$, then we say that t is collapsed.

Proposition 1: All alien subterms of a collapsed term are collapsed.

Proof: The proof can be found in Appendix A. ■

Lemma 1: Let $t_1 = C[[s_1, \dots, s_k]]$ and $t_2 = D[[s'_1, \dots, s'_l]]$ be two collapsed terms. Let $k \in \{a, b\}$ be such that $\text{root}(C) \in \mathcal{F}_k$. If $\text{root}(D) \notin \mathcal{F}_k$ then $t_1 \not\equiv_{E_a \cup E_b} t_2$. If $\text{root}(D) \in \mathcal{F}_k$, then:

$$t_1 \equiv_E t_2 \text{ iff } t_1 \{s_j \mapsto n_{[s_j]=E}\} \equiv_E t_2 \{s_j \mapsto n_{[s_j]=E}\}$$

where $n_{[s_j]=E}$ are names not appearing in t_1 or t_2 .

Proof:

The proof follows from Theorem 9.4.2, Chapter 9 (Combination Problems), Page 216 from [22]. ■

Lemma 2: If $s_1, \dots, s_k, s'_1, \dots, s'_l$ are collapsed terms with the root symbols from \mathcal{F}_c and C, D are both pure \mathcal{F}_c -contexts, we have that

$$C[s_1, \dots, s_k] \equiv_E D[s'_1, \dots, s'_l]$$

iff

$$C[n_{[s_1]=E}, \dots, n_{[s_k]=E}] \equiv_E D[n_{[s'_1]=E}, \dots, n_{[s'_l]=E}]$$

Proof: By case analysis on the definition of *collapse* and application of Lemma 1. ■

Lemma 3: If t_1, t_2 are collapsed terms such that $t_1 =_{\mathbb{E}} t_2$, we have that $\text{root}(t_1) \in \mathcal{F}_a$ iff $\text{root}(t_2) \in \mathcal{F}_a$.

Proof:

Immediate by Lemma 1. ■

C. Process algebra

Definition 5: Processes are defined inductively as follows:

$P, Q, R \dots$::=	processes
		0
		νx for $x \in \mathcal{X}$
		$\mathbf{in}(x)$ for $x \in \mathcal{X}$
		$(x := t)$ for $x \in \mathcal{X}, x \notin \text{vars}(t),$ $t \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}, \mathcal{X})$
		out (t) for $t \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}, \mathcal{X})$
		$[s = t]$ for $s, t \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}, \mathcal{X})$
		$(P \cdot Q)$
		$(P \mid Q)$
		$!P$

The process **0** does nothing. The process νx binds x to a fresh name. The process $\mathbf{in}(x)$ reads a term t from the public channel, and binds x to t . The assignment process $(x := t)$ instantiates x with t . The process $\mathbf{out}(t)$ outputs the term t on the public channel. The test process $[s = t]$ blocks if $s \neq_{\mathbb{E}} t$ and does nothing otherwise. The sequential composition process $P \cdot Q$ executes P followed by Q . The parallel composition process $(P \mid Q)$ runs P and Q in parallel. The replication process $!P$ will act as an infinite number of P s in parallel. We may write $\nu\{x_1, \dots, x_k\}$ instead of $\nu x_1 \cdot \dots \cdot \nu x_k$.

When we write processes, we assume that $!$ binds strongest, followed by $|$ and then by \cdot . We assume that the variables introduced by νx , $\mathbf{in}(x)$ and $(x := t)$ are bound throughout sequential composition \cdot as far to the right as possible. We identify processes up to α -renaming of bound variables and up to the following structural equivalence rules:

$$\begin{aligned}
P &\equiv P \cdot \mathbf{0} \equiv \mathbf{0} \cdot P \\
! \mathbf{0} &\equiv \mathbf{0} \\
P \mid \mathbf{0} &\equiv P \\
P \mid Q &\equiv Q \mid P \\
(P \mid Q) \mid R &\equiv P \mid (Q \mid R) \\
!P &\equiv P \mid !P
\end{aligned}$$

The operational semantics is given by the transition relations \rightarrow and \xrightarrow{t} (where t a term) described in Figure 1.

In the above semantics, \S is a meta-variable denoting either the empty string, in which case $\xrightarrow{\S} = \rightarrow$, or a recipe r , in which case $\xrightarrow{\S} = \xrightarrow{r}$.

The relation $\xrightarrow{t_1 \dots t_n}$ is defined as the reflexive and transitive closure of $\rightarrow \cup \xrightarrow{t}$.

Example 3: Continuing Example 1, $P_{\text{DH}} = \nu x_k \cdot (P_1 \mid P_2)$ models for the Diffie-Hellman protocol where

$$\begin{aligned}
\text{NEW} &\frac{P = \nu x \cdot R \quad x \notin \text{dom}(\sigma) \quad n \text{ fresh}}{(P, \varphi, \sigma) \rightarrow (R, \varphi, \sigma \cup \{x \mapsto n\})} \\
\text{INPUT} &\frac{P = \mathbf{in}(x) \cdot R \quad \varphi \vdash^r t \quad x \notin \text{dom}(\sigma)}{(P, \varphi, \sigma) \xrightarrow{t} (R, \varphi, \sigma \cup \{x \mapsto t\})} \\
\text{ASSGN} &\frac{P = (x := t) \cdot R \quad x \notin \text{dom}(\sigma) \quad \text{vars}(t) \subseteq \text{dom}(\sigma)}{(P, \varphi, \sigma) \rightarrow (R, \varphi, \sigma \cup \{x \mapsto t\})} \\
\text{OUTPUT} &\frac{P = \mathbf{out}(t) \cdot R \quad \text{vars}(t) \subseteq \text{dom}(\sigma)}{(P, \varphi, \sigma) \rightarrow (R, \varphi \cup \{w_{|\text{dom}(\varphi)|+1} \mapsto t\}, \sigma)} \\
\text{TEST} &\frac{P = [s = t] \cdot R \quad s\sigma =_{\mathbb{E}} t\sigma \quad \text{vars}(s, t) \subseteq \text{dom}(\sigma)}{(P, \varphi, \sigma) \rightarrow (R, \varphi, \sigma)} \\
\text{PARALLEL} &\frac{P = (Q_0 \mid Q_1) \cdot R \quad (Q_0, \varphi, \sigma) \xrightarrow{\S} (Q'_0, \varphi', \sigma')}{(P, \varphi, \sigma) \xrightarrow{\S} ((Q'_0 \mid Q_1) \cdot R, \varphi', \sigma')}
\end{aligned}$$

Fig. 1. Operational semantics.

$$\begin{aligned}
P_{\text{DH}}^1 &= \nu x \cdot \mathbf{out}(g(x)) \cdot \mathbf{out}(\text{mac}(g(x), x_k)) \cdot \mathbf{in}(z) \cdot \\
&\quad \mathbf{in}(z') \cdot [z' = \text{mac}(z, x_k)] \cdot y_1 := f(x, z) \\
P_{\text{DH}}^2 &= \nu y \cdot \mathbf{out}(g(y)) \cdot \mathbf{out}(\text{mac}(g(y), x_k)) \cdot \mathbf{in}(z) \cdot \\
&\quad \mathbf{in}(z') \cdot [z' = \text{mac}(z, x_k)] \cdot y_2 := f(y, z)
\end{aligned}$$

The process P_{DH}^1 models the first participant in an authenticated Diffie-Hellman key exchange while P_{DH}^2 models the second participant. The free variable x_k should be previously shared by the two participants to ensure authentication.

Examples of process executions can be found in the next section.

We conclude this section by defining secrecy and freshness. Intuitively, a variable is secret if in any protocol execution, its instantiation remains not deducible.

Definition 6 (secrecy): We say that a process P preserves the secrecy of $x \in \text{vars}(P)$ in the equational theory \mathbb{E} , and we denote it by $P \models_{\mathbb{E}} \text{Secret}(x)$, if whenever

$$(P, \emptyset, \emptyset) \xrightarrow{t_1 \dots t_n}^* (Q, \varphi, \sigma)$$

we have that $\varphi \not\vdash_{\mathbb{E}} x\sigma$.

Definition 7 (freshness): We say that a process P guarantees the freshness of $x \in \text{vars}(P)$ w.r.t. $\{y_1, \dots, y_k\} \subseteq \text{vars}(P)$ in the equational theory \mathbb{E} if whenever

$$(P, \emptyset, \emptyset) \xrightarrow{t_1 \dots t_n}^* (Q, \varphi, \sigma)$$

we have that $x\sigma \neq_{\mathbb{E}} y_i\sigma$ for all $1 \leq i \leq k$.

In the above definitions we assume that the variables in the processes have been conveniently α -renamed before application of the definition. If the above definitions concern variables appearing under replications, we assume that the conditions hold for any of the variables denoted by x (and

resp. y_1, \dots, y_k). This can be achieved formally by coloring all bound variables with different colors; whenever a variable is α -renamed it preserves its color. We would then say that a *certain color c is secret* when all variables colored with c remain secret. As this technicality is not essential in our approach, we prefer to use the less formal version.

III. COMPOSING PROCESSES

A. Difficulties

Of course, two protocols P, Q cannot in general be (securely) composed in arbitrary ways. We illustrate several cases where composing two secure protocols yields an attack. For readability, in this section we use the notation $\{s\}_t$ for the term $\text{enc}(s, t)$.

Revealing shared keys.

If a protocol P is establishing a (secret) key k , then a protocol Q using the key k should not reveal it. This would clearly compromise the security of P but it could also compromise the security of Q .

Assume for example that a protocol P_1 (playing the role of P above) generates two fresh (secret) data x and y and reveals the encryption of x under y :

$$P_1 = \nu x \cdot \nu y \cdot \mathbf{out}(\{x\}_y)$$

Note that P_1 may compute x and y in a more complicated way, but we consider just the rather trivial case where x and y are instantiated by fresh nonces for the sake of clarity. Then P_1 preserves the secrecy of both x and y . Assume now that a process Q_1 (playing the role of Q above) reveals y and uses x for encrypting a secret z :

$$Q_1 = \nu z \cdot \mathbf{out}(y) \cdot \mathbf{out}(\{z\}_x) \cdot \mathbf{0}$$

Then $\nu x' \cdot \nu y' \cdot (x := x') \cdot (y := y') \cdot Q_1$ preserves the secrecy of z , while the composition $P_1 \cdot Q_1$ of both processes does not preserve the secrecy of z . Indeed

$$(P_1 \cdot Q_1, \emptyset, \emptyset) \rightarrow^* (\mathbf{0}, \varphi, \sigma = \{x \mapsto k_1, y \mapsto k_2, z \mapsto k_3\})$$

where $\varphi = \{w_1 \mapsto \{k_1\}_{k_2}, w_2 \mapsto k_2, w_3 \mapsto \{k_3\}_{k_1}\}$. We have $\varphi \vdash_{\text{Enc}}^{\text{dec}(w_3, \text{dec}(w_1, w_2))} z\sigma$ and thus $P_1 \cdot Q_1 \not\vdash_{\text{Enc}} \text{Secret}(z)$.

Note that this attack works even if P_1 and Q_1 actually use different encryption symbols (thus even if they use disjoint signatures).

Sharing primitives.

The interaction of two protocols using common primitives may yield an attack, even if each of the protocols is secure when executed in isolation. Indeed, consider again the process P_1 described above and let Q_2 be a process that uses x for encrypting a secret z and outputs m for any m received under the encryption of y .

$$Q_2 = \nu z \cdot \mathbf{out}(\{z\}_x) \cdot \mathbf{in}(z') \cdot \mathbf{out}(\text{dec}(z', y)) \cdot \mathbf{0}$$

Then $\nu x' \cdot \nu y' \cdot (x := x') \cdot (y := y') \cdot Q_2$ preserves the secrecy of z , while the composition $P_1 \cdot Q_2$ of both processes does not preserve the secrecy of z . Indeed

$$(P_1 \cdot Q_2, \emptyset, \emptyset) \xrightarrow{*w_1} (\mathbf{0}, \varphi', \sigma)$$

where σ has been defined above and $\varphi' = \{w_1 \mapsto \{k_1\}_{k_2}, w_2 \mapsto \{k_3\}_{k_1}, w_3 \mapsto k_1\}$. We have $\varphi' \vdash_{\text{Enc}}^{\text{dec}(w_2, w_3)} z\sigma$, thus $P_1 \cdot Q_2 \not\vdash_{\text{Enc}} \text{Secret}(z)$.

So, in what follows, we will assume that the composed protocols use disjoint primitives. In Section V, we extend our result to the case where the protocols may share some primitives such as encryption and hash, provided they are tagged.

Key freshness.

It is important that shared variables (that are assumed to be fresh) are indeed instantiated by fresh values.

Assume for example that a protocol R is composed of three phases:

- it first generates a fresh key x : let $R_1 = \nu x$;
- it then runs a sub-protocol P to establish some secret y ;
- it outputs a fresh value z if $x = y$:
let $R_2 = \nu z \cdot [x = y] \cdot \mathbf{out}(z) \cdot \mathbf{0}$.

Then R is a secure protocol if y is a fresh key:

$$R_1 \cdot \nu k \cdot (y := k) \cdot R_2 \models \text{Secret}(z)$$

while it is not secure for all sub-protocols P establishing a secret key y . Indeed, let $P' = (y := x)$. Then $\nu k \cdot (x := k) \cdot P'$ preserves the secrecy of the shared key y but, because y is not fresh,

$$R_1 \cdot P' \cdot R_2 \not\vdash \text{Secret}(z)$$

In what follows, we will see that the counter-examples mentioned here are actually the only problematic cases. So we will require in our composition theorem that sub-protocols do not introduce equalities between shared variables.

B. Composition theorem

In order to state our composition theorem in a general way, we simply need to show that any execution trace on two combined processes can be transformed into an execution trace on one of the two processes. Then a trace of the composition leading to an attack can be transformed into a trace of one of the individual protocols leading to an attack.

Since an execution trace involves only a finite number of replications and determines the scheduling in parallel composition, we simply need to state our main result on *linear* processes, that is processes that contain neither parallel composition nor replication. We say that a process is *atomic* if it is linear and if it does not contain the sequencing operator \cdot . As illustrated in Section IV, our theorem can of course be applied to compose processes with arbitrary replications and parallel compositions.

Let $P = P_1 \dots P_n$ be a linear process over \mathcal{F}_a with free variables $\{x_1, \dots, x_p\}$ where P_i is an atomic process ($1 \leq i \leq n$). Let $Q = Q_1 \dots Q_m$ be a linear process over \mathcal{F}_b with free variables $\{y_1, \dots, y_q\}$ where Q_i is an atomic process ($1 \leq i \leq m$). Intuitively, the free variables of Q are established by P and conversely, the free variables of P are established by Q .

Let $R = R_1 \dots R_{n+m}$ be a *ground interleaving* of $P_1, \dots, P_n, Q_1, \dots, Q_m$, that is $\text{fv}(R) = \emptyset$ and $\{R_1, \dots, R_{n+m}\} = \{P_1, \dots, P_n, Q_1, \dots, Q_m\}$ (as multiset

equality). We consider R' a copy of R where the shared variables of P and Q are duplicated. More precisely, let $R' = R'_1 \dots R'_{n+m}$ be such that

- 1) $R'_i = P_j\{x \mapsto x^a\}$ if R_i is P_j for some j and where x ranges over all variables in P_j
- 2) $R'_i = Q_j\{x \mapsto x^b\}$ if R_i is Q_j for some j and where x ranges over all variables in Q_j

We consider an execution of the composition of P and Q .

$$(R, \emptyset, \emptyset) \xrightarrow{r_1 \dots r_k}^* (S_0, \varphi_0, \sigma_0) \xrightarrow{\S} (S, \varphi, \sigma) \quad (1)$$

where \S is a recipe r_{k+1} if the last action was an input and then empty string otherwise.

Assume w.l.o.g. that $x_1, \dots, x_{p'}$ are the variables from $\{x_1, \dots, x_p\}$ which appear in $\text{dom}(\sigma_0)$ and that $y_1, \dots, y_{q'}$ are the variables from $\{y_1, \dots, y_p\}$ which appear in $\text{dom}(\sigma_0)$. This means that $x_1, \dots, x_{p'}, y_1, \dots, y_{q'}$ are the shared variables which were instantiated before the last action of the execution.

Let $\{z_{[x_i\sigma_0]=E}\}$ and $\{z_{[y_i\sigma_0]=E}\}$ be fresh variables and let

$$\begin{aligned} R'' &= \nu\{z_{[x_i\sigma_0]=E}\}_{1 \leq i \leq p'} \cdot \nu\{z_{[y_i\sigma_0]=E}\}_{1 \leq i \leq q'} \\ x_1^a &:= z_{[x_1\sigma_0]=E} \cdot \dots \cdot x_{p'}^a := z_{[x_{p'}\sigma_0]=E} \\ y_1^b &:= z_{[y_1\sigma_0]=E} \cdot \dots \cdot y_{q'}^b := z_{[y_{q'}\sigma_0]=E} \\ R' & \end{aligned}$$

R'' corresponds to an interleaving of P and Q where P and Q do not share any variable anymore (since they are duplicated) and where the previously shared variables are instantiated by fresh distinct names. Note that whenever the execution of R instantiates two shared variables by the same value (e.g. $x_i\sigma = x_j\sigma$) then the (duplicated version of) x_i and x_j are instantiated in R'' by the same fresh name. This corresponds e.g. to the case where the same key is distributed among several participants (thus is assigned to distinct variables).

We are now ready to state our main theorem which says that we can mimic on R'' the execution trace of R (see Equation 1) unless P or Q do not preserve the secrecy or the freshness of the shared variables.

Theorem 1: Assume $\varphi_0 \not\vdash t$ for any $t \in \{x_1\sigma_0, \dots, x_{p'}\sigma_0, y_1\sigma_0, \dots, y_{q'}\sigma_0\}$ and that $x_i\sigma_0 \neq_E y_j\sigma_0$ for all $x_i, y_j \in \text{dom}(\sigma_0)$. Then there exist S', φ', σ' such that

$$(R'', \emptyset, \emptyset) \xrightarrow{r_1 \dots r_k \cdot \S}^* (S', \varphi', \sigma') \quad (2)$$

and

- 1) if $\varphi \vdash t$ for some $t \in \{x_i\sigma, y_j\sigma \mid x_i, y_j \in \text{dom}(\sigma)\}$, then $\varphi' \vdash s$ for some $s \in \{x_i^a\sigma', x_i^b\sigma', y_j^a\sigma', y_j^b\sigma' \mid x_i, y_j \in \text{dom}(\sigma)\}$
- 2) if there exist $x_i, y_j \in \text{dom}(\sigma)$ such that $x_i\sigma =_E y_j\sigma$ then $x_i^c\sigma' =_E y_j^c\sigma'$ for some $c \in \{a, b\}$
- 3) otherwise, if $\varphi \vdash x\sigma$ for some variable $x \in \text{vars}(P) \cap \text{dom}(\sigma)$ (resp. $\text{vars}(Q) \cap \text{dom}(\sigma)$), then $\varphi' \vdash x^a\sigma'$ (resp. $\varphi' \vdash x^b\sigma'$).

Intuitively, R'' is a composition of P and Q where the two process do not share variables anymore. Theorem 1 says that we can mimic on R'' the execution trace of R unless R reveals a shared variable, in which case R'' reveals some (duplicated)

shared variable as well. It will be used in the next section to conclude that one of the two processes P or Q (executed alone) reveals one of its (shared) variables thus is not secure. Indeed, assume w.l.o.g. that R'' reveals the variable x_i^a . Since R'' corresponds to P executed in parallel (and independently) of Q , the process Q can be entirely simulated by the adversary thus P reveals x_i , that is $P \not\vdash_E \text{Secret}(x_i)$.

The second condition is important to be able to completely separate the processes P and Q in R'' : as the variable x_i^a ($1 \leq i \leq p'$) and y_i^b ($1 \leq i \leq q'$) are instantiated in the process R'' by fresh names such that x_i^a and y_j^b receive the same name if $x_i\sigma =_E y_j\sigma$, it is important that such an equality does not happen. Otherwise, the two processes still share data and therefore we cannot conclude about either of them individually.

We now prove Theorem 1:

Proof:

Consider w.l.o.g. that σ is in collapsed form, i.e. $\sigma(x) = \text{collapse}(\sigma(x))$ for all $x \in \text{dom}(\sigma)$.

Let $n_{[x_i\sigma]=E}, n_{[y_j\sigma]=E}$ ($1 \leq i \leq p, 1 \leq j \leq q$) be fresh names. Let V_c ($c \in \{a, b\}$) be functions on terms defined as follows:

- 1) $V_c(t) = n_{[t]=E}$, if $t =_E s$ for some $s \in \{x_1\sigma_0, \dots, x_{p'}\sigma_0, y_1\sigma_0, \dots, y_{q'}\sigma_0\}$ and $\text{root}(t) \notin \mathcal{F}_c$
- 2) $f(V_d(t_1), \dots, V_d(t_k))$ if $f \in \mathcal{F}_d$, otherwise

The purpose of V_a (resp. V_b) is to replace the keys created by the process P (resp. Q) with fresh names such that equalities between terms are preserved. We have that V_c ($c \in \{a, b\}$) preserve equalities between terms:

Claim 1: If t_1, t_2 are collapsed terms such that $t_1 =_E t_2$, then $V_c(t_1) =_E V_c(t_2)$ ($c \in \{a, b\}$).

Proof: The proof can be found in Appendix B. \blacksquare

We will now use the functions V_c ($c \in \{a, b\}$) to construct the run in Equation 2 from the run in Equation 1.

Let σ' be defined as follows:

- 1) $\sigma'(x^a) = V_a(\sigma(x))$
- 2) $\sigma'(x^b) = V_b(\sigma(x))$
- 3) $\sigma'(z_{[x_i\sigma_0]=E}) = n_{[x_i\sigma_0]=E}$ ($1 \leq i \leq p'$)
- 4) $\sigma'(z_{[y_i\sigma_0]=E}) = n_{[y_i\sigma_0]=E}$ ($1 \leq i \leq q'$)

Assume $\varphi = \{w_1 \mapsto t_1\sigma, \dots, w_l \mapsto t_l\sigma\}$ (where $w_i \mapsto t_i\sigma$ comes from an atomic process $R_j = \text{out}(t_i)$). If R_j comes from P we let $t'_i = t_i\{x \mapsto x^a\}$ and if R_j comes from Q we let $t'_i = t_i\{x \mapsto x^b\}$ (where x ranges over all variables in t_i). We define $\varphi' = \{w_1 \mapsto t'_1\sigma', \dots, w_l \mapsto t'_l\sigma'\}$.

We proceed in two phases. In *Phase 1*, we show that Equation 2 holds. In *Phase 2*, we show that the Items 1, 2 and 3 from the conclusion are true.

Phase 1. We prove that the transition in Equation 2 holds for the σ' and φ' that we have defined and for some S' we do not care about. We prove this by induction on the number of transitions in Equation 2. For each transition we do a case-by-case analysis:

- 1) (tests work) if the transition is a test $[M = N]$ in R'' coming from P or Q . We assume w.l.o.g. that it comes from P (otherwise conclude by symmetry). Then M and N are pure a -terms. We have to prove that $M'\sigma' =_E N'\sigma'$, where $M' = M\{x \mapsto x^a\}$, $N' = N\{x \mapsto x^a\}$

and where x ranges over variables in M and respectively N .

By definition of M' , N' , σ' and V_a we have that $V_a(M\sigma) = M'\sigma'$ and $V_a(N\sigma) = N'\sigma'$.

Let $M\sigma = C[s_1, \dots, s_u]$ and $N\sigma = D[t_1, \dots, t_v]$ where C, D are a -contexts and $s_1, \dots, s_u, t_1, \dots, t_v$ are b -terms (C, D, s_i, t_j are uniquely determined – C and D might be $_$).

Because C and D are pure \mathcal{F}_a -contexts, they contain all of M and respectively N and therefore it follows that s_i and t_j must be subterms of σ . Therefore s_i and t_j are collapsed by Proposition 1. As $C[s_1, \dots, s_u] = M\sigma =_{\text{E}} N\sigma = D[t_1, \dots, t_v]$, we can apply Lemma 2 to conclude that

$$C[n_{[s_1]=\text{E}}, \dots, n_{[s_u]=\text{E}}] =_{\text{E}} D[n_{[t_1]=\text{E}}, \dots, n_{[t_v]=\text{E}}] \quad (3)$$

But $M'\sigma' = V_a(M\sigma) = C[V_a(s_1), \dots, V_a(s_u)]$ and $N'\sigma' = V_b(N\sigma) = D[V_a(t_1), \dots, V_a(t_v)]$ and, by Claim 1, $s_i =_{\text{E}} s_j$ (resp. $t_i =_{\text{E}} t_j$, resp. $s_i =_{\text{E}} t_j$) implies $V_a(s_i) =_{\text{E}} V_a(s_j)$ (resp. $V_a(t_i) =_{\text{E}} V_a(t_j)$, resp. $V_a(s_i) =_{\text{E}} V_a(t_j)$). Therefore, by applying the substitution $\tau = \{n_{[t]=\text{E}} \mapsto V_a(t)\}$, where t ranges over $\{s_1, \dots, s_u, t_1, \dots, t_v\}$, to Equation 3, we obtain

$$C[V_a(s_1), \dots, V_a(s_u)] =_{\text{E}} D[V_a(t_1), \dots, V_a(t_v)]$$

which is exactly $M'\sigma' =_{\text{E}} N'\sigma'$.

- 2) (inputs work) if the transition is the i -th input $\mathbf{in}(x)$ in R'' (coming from P (resp. Q)) we prove that $x^a\sigma' =_{\text{E}} r_i\varphi'$ (resp. $x^b\sigma' =_{\text{E}} r_i\varphi'$), where r_i is the i -th recipe on the $r_1 \dots r_k$ transition in Equation 2.

Assume w.l.o.g. that the i -th input in R'' comes from P (otherwise conclude by symmetry). We know that $x\sigma =_{\text{E}} r_i\varphi$ and we have to prove that $x^a\sigma' =_{\text{E}} r_i\varphi'$.

Definition 8: We say that a collapsed term t is *good* if one of the following conditions holds inductively:

- a) t is a subterm of σ
- b) t is deducible from φ and $t = E[[t_1, \dots, t_k]]$ for some pure context E and good terms t_1, \dots, t_k

Let $c \in \{a, b\}$. Let r be a recipe over φ such that for all strict (i.e. not itself) subterms r' of r we have that $r'\varphi \notin_{\text{E}} \{x_1, \dots, x_{p'}, y_1, \dots, y_{q'}\}\sigma_0$. Assume that $r\varphi \notin_{\text{E}} \{x_1, \dots, x_{p'}, y_1, \dots, y_{q'}\}\sigma_0$ or that $\text{root}(r\varphi) \in \mathcal{F}_c$. We show by structural induction on (the pure layers of) r that

Claim 2: We have that

$$V_c(r\varphi) =_{\text{E}} V_c(\text{collapse}(r\varphi))$$

and that $\text{collapse}(r\varphi)$ is a good term.

Proof: The proof can be found in Appendix B. ■

We are now ready to show that “inputs work”, namely that $x\sigma =_{\text{E}} r_i\varphi$ implies

$$x^a\sigma' =_{\text{E}} r_i\varphi' \quad (4)$$

As the input we are handling is one of the transitions in Equation 1, it follows that r_i is a recipe over φ_0 (because there is no transition after φ). Therefore, by

the hypothesis, $r_i\varphi = r_i\varphi_0$ cannot be a shared secret in (and neither can the subrecipes of r_i instantiated by φ) in $\{x_1, \dots, x_{p'}, y_1, \dots, y_{q'}\}\sigma_0$.

By the definition of σ' and φ' , we have that

$$x^a\sigma' = V_a(x\sigma) \quad (5)$$

and that

$$r_i\varphi' = V_a(r_i\varphi) = V_b(r_i\varphi) \quad (6)$$

In Equation 6, it does not matter if we have an a or a b as the subscript of V since $r_i\varphi$ cannot be a shared secret in $\{x_1, \dots, x_{p'}, y_1, \dots, y_{q'}\}\sigma_0$.

From our hypothesis

$$x\sigma =_{\text{E}} r_i\varphi$$

and as $r_i\varphi =_{\text{E}} \text{collapse}(r_i\varphi)$ we have that

$$V_a(x\sigma) =_{\text{E}} V_a(\text{collapse}(r_i\varphi))$$

by Claim 1. But combining this last equality with

$$V_a(\text{collapse}(r_i\varphi)) =_{\text{E}} V_a(r_i\varphi)$$

(which is immediate by Claim 2), we obtain

$$V_a(x\sigma) =_{\text{E}} V_a(r_i\varphi)$$

We combine this with Equations 5 and 6 to immediately derive Equation 4, which concludes the proof of this item.

- 3) (news work) if the transition is some νx in R'' , we have that $\sigma'(x)$ is a fresh name.

For processes νx in R'' but not in R , we have that $x\sigma' = n_-$ is a fresh name by the choice of n_- .

For processes νx^a in R' coming from P (for processes νx^b coming from Q the proof is analogous) we distinguish two cases:

- a) either x^a is a shared variable from $\{y_1, \dots, y_{q'}\}$, in which case $x^a\sigma$ is a name from \mathcal{F}_a . Therefore $x^a\sigma' = x\sigma$ by definition of σ' . We know that $x^a\sigma' = x\sigma$ is fresh with respect to the names n_- (by choice of n_-) and is fresh with respect with the other names by Equation 1.
- b) or x^a is not a shared variable, in which case $x^a\sigma' = x\sigma$ is fresh by the same reasoning.

- 4) (outputs work) if the transition is the i th $\mathbf{out}(t)$ in R'' , we have that $w_{i+1}\varphi' = t'\sigma'$. This is immediate by definition of φ' .

- 5) (old assignments work) if the transition is an assignment $x := t$ in R'' coming from P (resp. Q) we have that $x^a\sigma' =_{\text{E}} (t\{y \mapsto y^a\})\sigma'$ (resp. $x^b\sigma' =_{\text{E}} (t\{y \mapsto y^b\})\sigma'$), where y ranges over all variables in t . Let $M = x$, $N = t$, $M' = x^a$ and $N' = t'$ (resp. $M' = x^b$ and $N' = x^b$). Then this proof can be seen as an instance of the proof for Item 1 (tests work).

- 6) (new assignments work) if the transition is an assignment $x := t$ in R'' that did not come from P or Q : We show that for $j \in \{1, \dots, p'\}$ (resp. $j \in \{1, \dots, p'\}$) (i.e. for all assignments $x_j^a := z_{[x_i\sigma_0]=\text{E}}$

(resp. $y_j^b := z_{[y_i\sigma_0]=\mathbb{E}}$) in R'' but not in R' , we have that $x_j^a\sigma' = n_{[x_j\sigma_0]=\mathbb{E}}$ (resp. $y_j^b\sigma' = n_{[y_j\sigma_0]=\mathbb{E}}$). It is sufficient to prove that $\text{root}(x_j\sigma_0) \in \mathcal{F}_b$ (resp. $\text{root}(y_j\sigma_0) \in \mathcal{F}_a$), since then, by the definition of V_c , we have $x_j^a\sigma' = n_{[x_j\sigma_0]=\mathbb{E}}$ (resp. $y_j^b\sigma' = n_{[y_j\sigma_0]=\mathbb{E}}$). We prove something stronger by induction on the transitions in Equation 1.

Let $(R, \varphi_1, \sigma_1) \xrightarrow{\S_1} (R_2, \varphi_2, \sigma_2) \xrightarrow{\S_2} \dots \xrightarrow{\S_n} (R_n, \varphi_n, \sigma_n)$, where $\sigma_1 = \emptyset$, $\varphi_1 = \emptyset$, $R_n = S$, $\varphi = \varphi_n$, $\sigma = \sigma_n$ and \S_i is either some recipe r_j or the empty string, be the transitions in Equation 1.

Definition 9: We say that a collapsed term t is i -good if one of the following conditions holds inductively:

- there exists $x \in \{x_1, \dots, x_{p'}, y_1, \dots, y_{q'}\} \cap \text{dom}(\sigma_i)$ such that $t = x\sigma$
- t is deducible from φ_i and $t = C[[t_1, \dots, t_k]]$ for some i -good terms t_1, \dots, t_k .

Claim 3: For $i \in \{0, \dots, n\}$, we have that $\text{collapse}(w_j\varphi_i)$ (for all $w_j \in \text{dom}(\varphi_i)$) is an i -good term and $x\sigma_i = C[s_1, \dots, s_k]$ (for all $x \in \text{dom}(\sigma_i)$) for a pure (possibly empty) \mathcal{F}_c -context C and $i-1$ -good terms s_1, \dots, s_k (with $\text{root}(s_j) \in \mathcal{F}_c$), where $c = a$ if $x \in \text{vars}(P) \setminus \{x_1, \dots, x_{p'}\}$ and $c = b$ if $x \in \text{vars}(Q) \setminus \{y_1, \dots, y_{q'}\}$.

Proof: The proof can be found in Appendix B. ■

Now it is easy to prove that $\text{root}(x_j\sigma) \in \mathcal{F}_b$ ($1 \leq j \leq p'$). Indeed, assume w.l.o.g. that $x_1, \dots, x_{p'}$ appear in σ in this order.

We prove by well founded induction on $1 \leq i \leq p'$ that $\text{root}(x_i\sigma) \in \mathcal{F}_b$. Suppose by contradiction that $\text{root}(x_i\sigma) \in \mathcal{F}_a$. Then, by Claim 3, we have that $x_i\sigma$ is a j -good term. As $x_i\sigma$ is not deducible from φ_0 (by hypothesis), it follows that we are in the first case of the definition of j -good and therefore there exists a shared variable x_j ($1 \leq j < i$) or y_j ($1 \leq j \leq q'$) appearing before x_i such that $x_i\sigma = x_j\sigma$ or $x_i\sigma = y_j\sigma$. But the second case is impossible by the hypothesis of the theorem (freshness of x_- w.r.t. y_-). Therefore $x_i\sigma = x_j\sigma$ with $j < i$ and therefore we obtained a contradiction (by applying the induction hypothesis, we have that $\text{root}(x_j\sigma) \in \mathcal{F}_b$).

Phase 2. We have shown in Phase 1 that Equation 2 holds for our definition of σ' and φ' . We show now that each item in the conclusion of the theorem holds.

- if $\varphi \vdash t$ for some $t \in \{x_1\sigma, \dots, x_{p'}\sigma, y_1\sigma, \dots, y_{q'}\sigma\}$, then $\varphi' \vdash t$ for some

$$s \in \{x_1^a\sigma', \dots, x_{p'}^a\sigma', x_1^b\sigma', \dots, x_{p'}^b\sigma', y_1^a\sigma', \dots, y_{q'}^a\sigma', y_1^b\sigma', \dots, y_{q'}^b\sigma'\}$$

Suppose r is a minimal recipe such that $r\varphi$ is a shared secret. Then none of the subrecipes of r , instantiated by φ , can be shared secrets in $\{x_1, \dots, x_{p'}, y_1, \dots, y_{q'}\}\sigma_0$. Assume w.l.o.g. that the shared secret is an $x_j\sigma =_{\mathbb{E}} r\varphi$. Let c be such that $\text{root}(r\varphi) \in \mathcal{F}_c$.

We know that $x_j^c\sigma' = V_c(x_j\sigma)$ by definition. We also know by Claim 1 that $V_c(x_j\sigma) =_{\mathbb{E}} V_c(\text{collapse}(r\varphi))$

(because $\text{collapse}(r\varphi) =_{\mathbb{E}} r\varphi =_{\mathbb{E}} x_j\sigma$). Therefore $x_j^c\sigma' =_{\mathbb{E}} V_c(\text{collapse}(r\varphi))$. But by Claim 2, we have that $V_c(r\varphi) =_{\mathbb{E}} V_c(\text{collapse}(r\varphi))$ and therefore $x_j^c\sigma' =_{\mathbb{E}} V_c(r\varphi)$. But $V_c(r\varphi)$ is, by the definition of φ' , equal to $r\varphi'$. Therefore we can chose $s = x_j^c\sigma' =_{\mathbb{E}} r\varphi'$ which is deducible and we conclude this item.

- if there exist i, j such that $x_i \in \text{dom}(\sigma)$ and $y_j \in \text{dom}(\sigma)$ and $x_i\sigma =_{\mathbb{E}} y_j\sigma$ then $x_i^c\sigma' =_{\mathbb{E}} y_j^c\sigma'$. By definition we have $x_i^c\sigma' = V_c(x_i\sigma)$ and $y_j^c\sigma' = V_c(y_j\sigma)$. We immediately conclude by Claim 1.
- otherwise, if $\varphi \vdash x\sigma$ for some variable $x \in \text{vars}(P)$ then $\varphi' \vdash x^a\sigma'$ (the case with Q is symmetric). Let r be a recipe for $x\sigma$. As no shared secret is deducible (otherwise we would be in the case of Item 1) we can apply Claim 2 to obtain that

$$V_c(r\varphi) =_{\mathbb{E}} V_c(\text{collapse}(r\varphi)).$$

By definition $x^a\sigma' = V_a(x\sigma)$. But $V_a(x\sigma) =_{\mathbb{E}} V_a(\text{collapse}(r\varphi)) =_{\mathbb{E}} V_a(r\varphi) = r\varphi'$. Therefore we obtain $x^a\sigma' =_{\mathbb{E}} r\varphi'$, which means $x^a\sigma'$ is deducible. ■

C. Some further useful lemmas

Theorem 1 is our key result for composing processes. We list here some other useful (and rather straightforward) results that we will use to show how to securely compose processes.

Theorem 1 is stated for *linear processes*. Given an arbitrary process P , we say that $Q_1 \dots Q_n$ is a *linearization* of P if $P \xrightarrow{Q_1} P_1 \xrightarrow{Q_2} \dots \xrightarrow{Q_n} P_n$ where the transitions $\xrightarrow{Q_i}$ are defined in Figure 2. Intuitively, a linearization of P is a symbolic partial trace of P .

We denote by $L(P)$ the set of linearizations of P . It intuitively consists of the set of all possible interleaving for the executions of P . We will use this set for reasoning about protocols containing parallel composition and replications.

$$\begin{array}{l} \text{NEW} \quad \frac{P = \nu x \cdot R}{P \xrightarrow{\nu x} R} \quad \text{INPUT} \quad \frac{P = \mathbf{in}(x) \cdot R}{P \xrightarrow{\mathbf{in}(x)} R} \\ \text{ASSGN} \quad \frac{P = (x := t) \cdot R}{P \xrightarrow{x := t} R} \quad \text{TEST} \quad \frac{P = [s = t] \cdot R}{P \xrightarrow{[s=t]} R} \\ \text{OUTPUT} \quad \frac{P = \mathbf{out}(t) \cdot R}{P \xrightarrow{\mathbf{out}(t)} R} \\ \text{PARALLEL} \quad \frac{P = (Q_0 \mid Q_1) \cdot R \quad Q_0 \xrightarrow{\S} Q'_0}{P \xrightarrow{\S} Q'_0 \mid Q_1} \end{array}$$

Fig. 2. Linearization.

A process P preserves a secret if and only if all its linearizations preserve the secret.

Lemma 4: Let P be a process. Then for any equational theory E , $P \models_E \text{Secret}(x)$ iff for all $Q \in L(P)$ we have that $Q \models_E \text{Secret}(x)$.

Proof:

By induction on the number of transitions and case analysis. ■

One can also notice that if a protocol reveals a secret then it *a fortiori* reveals it when projecting two names on a single one.

Lemma 5: For any equational theory E , if $\nu x_1 \cdot \nu x_2 \cdot P \not\models_E \text{Secret}(x)$ then $\nu x_1 \cdot x_2 := x_1 \cdot P \not\models_E \text{Secret}(x)$.

Proof:

By induction on the number of transitions in the trace leading to the revelation of x . ■

We also need to show that, when mounting an attack on a process P on the signature \mathcal{F}_a , the adversary is not more powerful when using the combined theory $E_a \cup E_b$. This is captured by the following lemma.

Lemma 6: If P is a linear process over \mathcal{F}_a and

$$(P, \emptyset, \emptyset) \xrightarrow{\S_1}_E (P_1, \varphi_1, \sigma_1) \xrightarrow{\S_2}_E \dots \xrightarrow{\S_n}_E (P_n, \varphi_n, \sigma_n)$$

then

$$(P, \emptyset, \emptyset) \xrightarrow{\S'_1}_{E_a} (P_1, \varphi'_1, \sigma'_1) \xrightarrow{\S'_2}_{E_a} \dots \xrightarrow{\S'_n}_{E_a} (P_n, \varphi'_n, \sigma'_n)$$

for some φ'_n, σ'_n and \S'_n such that $\varphi_n \vdash_E x\sigma_n$ implies $\varphi'_n \vdash_{E_a} x\sigma_n$ and $x\sigma_n =_E y\sigma_n$ implies $x\sigma'_n =_{E_a} y\sigma'_n$ for all $x, y \in \text{dom}(\sigma_n)$.

Note that this lemma relies on the assumption we made that E_b is not trivial (it does not equate all terms). Otherwise $E_a \cup E_b$ would be trivial and therefore all terms would be deducible. Even though the intuition behind this lemma is straightforward, the proof is rather technical (presented in Appendix B).

From the above lemma, we immediately obtain:

Corollary 1: If P is a process over \mathcal{F}_a and $P \models_{E_a} \text{Secret}(x)$, then $P \models_E \text{Secret}(x)$.

IV. APPLICATIONS

We now show how to apply our composition theorem in several contexts.

A. Key-exchange protocol

It is often the case that a security protocol is verified assuming that some keys are already shared between the principals, abstracting away from the process by which these keys have been established. We can use our result to show that if a key exchange protocol was used to establish a shared key and if the two protocols use disjoint cryptographic primitives, their composition is secure provided that neither the key exchange protocol nor the main protocol reveal the established keys.

To state our result, we first need the following definition:

Definition 10: We say that P binds x if $P = P_1 \cdot P_2 \cdot \dots \cdot P_n$ and $P_j \in \{\text{in}(x), x := t, \nu x\}$ for some $1 \leq j \leq n$ and some term t (note that P_1, \dots, P_n are not necessarily atomic).

Theorem 2: Let $P = \nu k_1 \cdot \dots \cdot \nu k_n \cdot (P_1 \mid P_2)$ be a process over \mathcal{F}_a and let $Q = \nu k \cdot (x_k := k \cdot Q_1 \mid y_k := k \cdot Q_2)$ be a process over \mathcal{F}_b such that:

- P_1 binds x_k and P_2 binds y_k
- $\text{fv}(P) = \emptyset, \text{fv}(Q) = \emptyset$ and $\text{vars}(P) \cap \text{vars}(Q) = \{x_k, y_k\}$
- $P \models_{E_a} \text{Secret}(x_k)$ and $P \models_{E_a} \text{Secret}(y_k)$
- $Q \models_{E_b} \text{Secret}(x_k)$ and $Q \models_{E_b} \text{Secret}(y_k)$.

If $Q \models_{E_b} \text{Secret}(x_s)$ then $W = \nu k_1 \cdot \dots \cdot \nu k_n \cdot (P_1 \cdot Q_1 \mid P_2 \cdot Q_2) \models_E \text{Secret}(x_s)$.

Intuitively, the protocol P corresponds to two roles P_1 and P_2 that establish a key k stored respectively in x_k for P_1 and in y_k for P_2 . Then each of the two roles Q_1 and Q_2 of Q uses respectively its version of the key. Theorem 2 ensures that the protocol P can be safely abstracted by the generation of a single fresh key, distributed among the participants.

This result could easily be extended to an arbitrary number of roles. Note that Q_1 and Q_2 may contain replications thus the key k may be used in several distinct sessions.

Proof:

If $c \in \{a, b\}$ and if P is a process, we denote by P^c the process in which any occurrence of a variable $x \in \text{vars}(P)$ has been replaced by the variable x^a .

We do a proof by contradiction. We assume that $W \not\models_E \text{Secret}(x_s)$. Then, by Lemma 4, we have that there exists a linearization R of W such that $R \not\models_E \text{Secret}(x_s)$.

R is then a ground interleaving of a linearization $P_0 \in L(P)$ and a linearization $Q_0 \in L(Q_1 \mid Q_2)$. We denote by R' the same ground interleaving of P_0^a and Q_0^b .

As $R \not\models_E \text{Secret}(x_s)$, there is a trace

$$(R, \emptyset, \emptyset) \xrightarrow{\S_1} (R_1, \varphi_1, \sigma_1) \xrightarrow{\S_2} \dots \xrightarrow{\S_m} (R_m, \varphi_m, \sigma_m)$$

such that $\varphi_m \vdash x_s \sigma_m$.

Let $1 \leq l \leq m$ be the first index such that $\varphi_l \vdash x_k \sigma_l$ or $\varphi_l \vdash y_k \sigma_l$ or $\varphi_l \vdash x_s \sigma_l$.

We can then apply Theorem 1 to obtain that the process

$$R'' = \nu\{z_a, z_b\} \cdot (x_k^b := z_a) \cdot (y_k^b := z_b) \cdot R'$$

reveals $x_k^a, y_k^a, x_k^b, y_k^b$ or x_s^b in the equational theory E . (We do not know if z_a and z_b are the same variable). In either case, by Lemma 5, we have that

$$R''' = \nu k^b \cdot (x_k^b := k^b) \cdot (y_k^b := k^b) \cdot R'$$

reveals $x_k^a, y_k^a, x_k^b, y_k^b$ or x_s^b .

But R''' is an interleaving of some linearization of P^a and Q^b . Therefore R''' is a linearization of $P^a \mid Q^b$.

Therefore $P^a \mid Q^b$ reveals $x_k^a, y_k^a, x_k^b, y_k^b$ or x_s^b . Assume $P^a \mid Q^b$ reveals x_k^a or y_k^a . Since P^a and Q^b share no data, Q^b can be simulated by the adversary and thus $P^a \not\models_E \text{Secret}(z)$ for some $z \in \{x_k^a, y_k^a\}$. If $P^a \mid Q^b$ reveals x_k^b, y_k^b or x_s^b , we similarly deduce that $Q^b \not\models_E \text{Secret}(z)$ for some $z \in \{x_k^b, y_k^b, x_s^b\}$. By Corollary 1, we obtain that $P \not\models_{E_a} \text{Secret}(z)$ for some $z \in \{x_k, y_k\}$ or that $Q \not\models_{E_b} \text{Secret}(z)$

for some $z \in \{x_k, y_k, x_s\}$. In both cases, this contradicts the hypotheses. We thus deduce that $W \models_{\text{E}} \text{Secret}(x_s)$. \blacksquare

B. Secure channels

Another composition scenario is when a protocol is proven secure assuming some secure channels, that is, assuming that some secret key is established on the fly. We show that the secure channel can be implemented by any sub-protocol provided that neither the main protocol nor the sub-protocol reveal the key.

Theorem 3: Let $P = \nu k_1 \dots \nu k_n \cdot (P_1 \mid P_2)$ be a process over \mathcal{F}_a and let $Q = (\nu k \cdot (x_k := k \cdot Q_1 \mid y_k := k \cdot Q_2))$ be a process over \mathcal{F}_b such that:

- 1) P_1 binds x_k and P_2 binds y_k
- 2) $fv(P) = fv(Q) = \emptyset$
- 3) $P \models_{\text{E}_a} \text{Secret}(x_k)$ and $P \models_{\text{E}_a} \text{Secret}(y_k)$
- 4) $Q \models_{\text{E}_b} \text{Secret}(x_k)$ and $Q \models_{\text{E}_b} \text{Secret}(y_k)$ and $Q \models_{\text{E}_b} \text{Secret}(x_s)$

Then $R = \nu k_1 \dots \nu k_n \cdot (P_1 \cdot Q_1 \mid P_2 \cdot Q_2) \models_{\text{E}} \text{Secret}(x_s)$.

Compared to Theorem 2, the two roles Q_1 and Q_2 now use a different key k in each session.

Proof:

We prove the result by contradiction along the same lines as the proof of Theorem 2. We assume that $W \not\models_{\text{E}} \text{Secret}(x_s)$. Then, by Lemma 4, we have that there exists a linearization R of W such that $R \not\models_{\text{E}} \text{Secret}(x_s)$.

R is then a ground interleaving of a linearization $P_0 \in \text{L}(P)$ and a linearization $Q_0 \in \text{L}(! (Q_1 \mid Q_2))$. We denote by R' the same ground interleaving of P_0^a and Q_0^b .

As $R \not\models_{\text{E}} \text{Secret}(x_s)$, there is a trace

$$(R, \emptyset, \emptyset) \xrightarrow{\S_1} (R_1, \varphi_1, \sigma_1) \xrightarrow{\S_2} \dots \xrightarrow{\S_m} (R_m, \varphi_m, \sigma_m)$$

such that $\varphi_m \vdash x_s \sigma_m$.

Let $1 \leq l \leq m$ be the first index such that $\varphi_l \vdash x_k \sigma_l$ or $\varphi_l \vdash y_k \sigma_l$ or $\varphi_l \vdash x_s \sigma_l$.

We can then apply Theorem 1 to obtain that the process

$$R'' = \nu z_a, z_b \cdot (x_k^b := z_a) \cdot (y_k^b := z_b) \cdot R'$$

reveals $x_k^a, y_k^a, x_k^b, y_k^b$ or x_s^b in the equational theory E. (We do not know if z_a and z_b are the same variable). In either case, by Lemma 5, we have that

$$R''' = \nu k^b \cdot (x_k^b := k^b) \cdot (y_k^b := k^b) \cdot R'$$

reveals $x_k^a, y_k^a, x_k^b, y_k^b$ or x_s^b .

But R''' is an interleaving of some linearization of P^a and Q^b . Therefore R''' is a linearization of $P^a \mid Q^b$.

Therefore $P^a \mid Q^b$ reveals $x_k^a, y_k^a, x_k^b, y_k^b$ or x_s^b . Assume $P^a \mid Q^b$ reveals x_k^a or y_k^a . Since P^a and Q^b share no data, Q^b can be simulated by the adversary and thus $P^a \not\models_{\text{E}} \text{Secret}(z)$ for some $z \in \{x_k^a, y_k^a\}$. If $P^a \mid Q^b$ reveals x_k^b, y_k^b or x_s^b , we similarly deduce that $Q^b \not\models_{\text{E}} \text{Secret}(z)$ for some $z \in \{x_k^b, y_k^b, x_s^b\}$. By Corollary 1, we obtain that $P \not\models_{\text{E}_a} \text{Secret}(z)$ for some $z \in \{x_k, y_k\}$ or that $Q \not\models_{\text{E}_b} \text{Secret}(z)$ for some $z \in \{x_k, y_k, x_s\}$. In both cases, this contradicts the hypotheses. We thus deduce that $W \models_{\text{E}} \text{Secret}(x_s)$. \blacksquare

Example 4: Consider the process P_{DH} defined in Example 3, over the signature \mathcal{F}_{DH} . Consider any other protocol

$$Q = \nu y_k \cdot (y_1 := y_k \cdot Q_1 \mid y_2 := y_k \cdot Q_2)$$

that models a protocol in which a participant Q_1 sends a secret x_s to the second participant using a shared key y_k . Assume that Q is defined over the signature \mathcal{F}_{enc} and that $Q \models_{\text{E}_{\text{enc}}} \text{Secret}(x_s)$. Then the sequential composition of P_{DH} and Q , where P_{DH} is used to establish the shared key used in Q is defined by

$$W = \nu x_k \cdot (P_{\text{DH}}^1 \cdot Q_1 \mid P_{\text{DH}}^2 \cdot Q_2)$$

Applying Theorem 2, $W \models_{\text{E}_{\text{DH}} \cup \text{E}_{\text{enc}}} \text{Secret}(x_s)$, that is W does not leak x_s in the theory $\text{E}_{\text{DH}} \cup \text{E}_{\text{enc}}$.

V. TAGGING

Tagging is a syntactic transformation of a protocol in order, for example, to make it more resistant against attacks. Many ways to tag protocols have been proposed in different contexts, e.g. for composing protocols [11], [13] as discussed in introduction, to facilitate their analysis [23], [24] or to prevent type-flow attacks [25]. Typically, tagging a security protocol consists in appending a tag (e.g. a number, a nonce or a protocol identifier) to each plaintext before encrypting it and removing the tag after decryption. Tagging a protocol does not introduce additional attacks in the protocol, while preserving its communication goals.

In the previous sections, we have shown how to compose protocols which do not share cryptographic primitives. In this section, we show that protocols which do share common cryptographic primitives, such as encryption and hash functions, can also be securely composed in the same manner, as long as the two protocols are tagged differently.

Our proof technique relies on our previous theorems, in that we show that an attack against the composition of two differently tagged protocols can be transformed into an attack where the protocols use different encryption and hash functions. Therefore, tagging essentially enforces the disjointness of the two protocols.

Even though we prove this only for symmetric encryption and hash functions, our technique can be extended in a straightforward manner to other usual cryptographic primitives such as asymmetric encryption and digital signature. Other cryptographic primitives can pose more problems (for example it is not obvious if/how the *exclusive OR* can be tagged). It is an interesting open problem to give a generic definition of tagging and characterize which cryptographic primitives can be tagged.

We consider protocols over the signature $\mathcal{F}_{\text{enc},h} = \{\text{enc}, \text{dec}, h\}$ where *enc* and *dec* model respectively encryption and decryption and are of arity 2 and *h* models a hash function and is of arity 1. We also consider the signatures $\mathcal{F}_{\text{enc},h}^a = \{\text{enc}_a, \text{dec}_a, h_a\}$ and $\mathcal{F}_{\text{enc},h}^b = \{\text{enc}_b, \text{dec}_b, h_b\}$. We consider the associated equational theory E_{enc} defined in Example 2 and the equational theories $\text{E}_{\text{enc}}^a = \{\text{dec}_a(\text{enc}_a(x, y), y) = x\}$

and $E_{\text{enc}}^b = \{dec_b(enc_b(x, y), y) = x\}$. The signatures $\mathcal{F}_{\text{enc}, h}^a$ and $\mathcal{F}_{\text{enc}, h}^b$, together with the associated equational theories E_{enc}^a and E_{enc}^b , can be considered to intuitively model different implementations of the encryption/decryption/hash functions.

In order to define tagging, we first consider the signature renaming transformation $_c$ ($c \in \{a, b\}$) which assigns to a protocol P over $\mathcal{F}_{\text{enc}, h}$ a protocol P^c ($c \in \{a, b\}$) in the signature $\mathcal{F}_{\text{enc}, h}^c$ such that the two protocols are identical modulo bijective renaming of functions symbols (enc , dec and h are transformed into enc_c , dec_c and respectively h_c and this transformation is extended homomorphically to the entire protocol).

Example 5: If $P = \nu y \cdot \mathbf{in}(x) \cdot \mathbf{out}(dec(x, y))$, then $P^a = \nu(y) \cdot \mathbf{in}(x) \cdot \mathbf{out}(dec_a(x, y))$.

For $c \in \{a, b\}$ we consider a *tagging function symbol* tag_c and an *untagging function symbol* $untag_c$ contained in the signature $\mathcal{F}_c = \{tag_c, untag_c\}$ (where both function symbols have arity 1). The role of the tag_c function is to *tag* its argument with the tag c . Typically, this means appending c to the argument but the precise implementation of the tagging function does not need to be specified. The role of the $untag_c$ function is to remove the tag. To model this interaction between tag_c and $untag_c$ we consider the equational theories $E_c = \{untag_c(tag_c(x)) = x\}$ (for $c \in \{a, b\}$).

If $A \in \{\mathbf{in}(x), \mathbf{out}(t), \nu x, x := t, s = t\}$ is an atomic action over $\mathcal{F}_{\text{enc}, h}^c$ (with $c \in \{a, b\}$), we let $\llbracket A \rrbracket$ be a linear process over $\mathcal{F}_{\text{enc}, h} \cup \mathcal{F}_c$ denoting the c -tagged version of A , defined as follows:

$$\begin{aligned} \llbracket \mathbf{in}(x) \rrbracket &= \mathbf{in}(x) \\ \llbracket \mathbf{out}(t) \rrbracket &= tests^c(\mathcal{H}(t)) \cdot \mathbf{out}(\mathcal{H}(t)) \\ \llbracket \nu x \rrbracket &= \nu x \\ \llbracket x := t \rrbracket &= tests^c(\mathcal{H}(t)) \cdot x := \mathcal{H}(t) \\ \llbracket s = t \rrbracket &= tests^c(\mathcal{H}(s)) \cdot tests^c(\mathcal{H}(t)) \cdot x := \mathcal{H}(t) \end{aligned}$$

where $\mathcal{H}(t)$ tags the term t with c as defined below:

$$\begin{aligned} \mathcal{H}(enc_c(t_1, t_2)) &= enc(tag_c(\mathcal{H}(t_1)), \mathcal{H}(t_2)) \\ \mathcal{H}(dec_c(t_1, t_2)) &= untag_c(dec(\mathcal{H}(t_1), \mathcal{H}(t_2))) \\ \mathcal{H}(h_c(t_1)) &= h(tag_c(\mathcal{H}(t_1))) \\ \mathcal{H}(u) &= u \quad \text{if } u \text{ is a name or a variable} \end{aligned}$$

and where $tests^c(t)$ is a sequence of tests which ensure that every decryption and every untagging performed by the protocol is successful:

$$\begin{aligned} tests^c(enc(t_1, t_2)) &= tests^c(t_1) \cdot tests^c(t_2) \\ tests^c(h(t_1)) &= tests^c(t_1) \\ tests^c(tag_c(t_1)) &= tests^c(t_1) \\ tests^c(dec(t_1, t_2)) &= [enc(dec(t_1, t_2), t_2) = t_1] \cdot \\ &\quad tests^c(t_1) \cdot tests^c(t_2) \\ tests^c(untag_c(t_1)) &= [tag_c(untag_c(t_1)) = t_1] \cdot tests^c(t_1) \\ tests^c(u) &= \mathbf{0} \quad \text{if } u \text{ is a name or a variable} \end{aligned}$$

The transformation $\llbracket _ \rrbracket$ is extended homomorphically to composed processes.

Example 6: Continuing Example 5, we have that

$$\mathcal{H}(dec_a(x, y)) = untag_a(dec(x, y))$$

and that

$$\begin{aligned} tests^a(untag_a(dec(x, y))) &= \\ [tag_a(untag_a(dec(x, y))) = dec(x, y)] \cdot \\ [enc(dec(x, y), y) = x]. \end{aligned}$$

Finally, we have that

$$\begin{aligned} \llbracket P^a \rrbracket &= \nu y \cdot \mathbf{in}(x) \cdot [tag_a(untag_a(dec(x, y))) = dec(x, y)] \cdot \\ &\quad [enc(dec(x, y), y) = x] \cdot \mathbf{out}(untag_a(dec(x, y))). \end{aligned}$$

Note that before performing the decryption, the process $\llbracket P^a \rrbracket$ verifies that the received term is a valid encryption and that the underlying plain-text has been correctly tagged.

Our next lemma shows why tagging two protocols is essentially the same as forcing them to use disjoint equational theories. It allows us to reduce the security problem for differently tagged processes to the security problem for processes which use disjoint equational theories. It states that if there is an attack on a composition of two differently tagged protocols $\llbracket P^a \rrbracket$ and $\llbracket Q^b \rrbracket$, there is an attack on the composition of the same protocols before tagging (P^a and Q^b), where the encryption and hash functions come from disjoint equational theories.

Lemma 7: Let P and Q be linear processes over $\mathcal{F}_{\text{enc}, h}$. Let W be an arbitrary interleaving of P^a and Q^b and let $R = \llbracket W \rrbracket$. If R reveals x then W reveals x .

Proof:

The technique is to transform a trace leading to an attack on R into a trace leading to an attack on W . The proof can be found in Appendix C. ■

Furthermore, we have that if a protocol is secure then its c -tagged version is secure.

Lemma 8: If P is a protocol over $\mathcal{F}_{\text{enc}, h}$, then $P \models_{E_{\text{enc}}} \text{Secret}(x)$ implies $\llbracket P^c \rrbracket \models_{E_{\text{enc}} \cup E_c} \text{Secret}(x)$.

The proof of this lemma can be found in Appendix C.

We can now state a generic theorem, in the spirit of Theorem 1, but for tagged protocols.

Let $P = P_1 \dots P_n$ be a linear process over $\mathcal{F}_{\text{enc}, h}$ with free variables $\{x_1, \dots, x_p\}$ where P_i is an atomic process ($1 \leq i \leq n$). Let $Q = Q_1 \dots Q_m$ be a linear process over $\mathcal{F}_{\text{enc}, h}$ with free variables $\{y_1, \dots, y_q\}$ where Q_i is an atomic process ($1 \leq i \leq m$).

Let $R = R_1 \dots R_{n+m}$ be a ground interleaving of $\llbracket P_1^a \rrbracket, \dots, \llbracket P_n^a \rrbracket, \llbracket Q_1^b \rrbracket, \dots, \llbracket Q_m^b \rrbracket$. We consider R' a untagged copy of R where the shared variables of P and Q are duplicated as in Theorem 1 such that P and Q access disjoint variables. More precisely, let $R' = R'_1 \dots R'_{n+m}$ be such that:

- 1) $R'_i = P_j\{x \mapsto x^a\}$ if R_i is $\llbracket P_j^a \rrbracket$ for some j and where x ranges over all variables in P_j
- 2) $R'_i = Q_j\{x \mapsto x^b\}$ if R_i is $\llbracket Q_j^b \rrbracket$ for some j and where x ranges over all variables in Q_j

We consider an execution of the composition of $[[P^a]]$ and $[[Q^b]]$ in the equational theory $E_{\text{enc}} \cup E_a \cup E_b$.

$$(R, \emptyset, \emptyset) \xrightarrow{r_1 \dots r_k} (S_0, \varphi_0, \sigma_0) \xrightarrow{\S} (S, \varphi, \sigma) \quad (7)$$

where \S is a recipe r_{k+1} if the last action was an input and then empty string otherwise.

Assume w.l.o.g. that $x_1, \dots, x_{p'}$ are the variables from $\{x_1, \dots, x_p\}$ which appear in $\text{dom}(\sigma_0)$ and that $y_1, \dots, y_{q'}$ are the variables from $\{y_1, \dots, y_p\}$ which appear in $\text{dom}(\sigma_0)$. This means that $x_1, \dots, x_{p'}, y_1, \dots, y_{q'}$ are the shared variables which were instantiated before the last action of the execution.

Let $\{z_{[x_i\sigma_0]=E}\}$ and $\{z_{[y_i\sigma_0]=E}\}$ be fresh variables and let

$$\begin{aligned} R'' &= \nu \{z_{[x_i\sigma_0]=E}\}_{1 \leq i \leq p'} \cdot \nu \{z_{[y_i\sigma_0]=E}\}_{1 \leq i \leq q'} \cdot \\ & x_1^a := z_{[x_1\sigma_0]=E} \cdot \dots \cdot x_{p'}^a := z_{[x_{p'}\sigma_0]=E} \cdot \\ & y_1^b := z_{[y_1\sigma_0]=E} \cdot \dots \cdot y_{q'}^b := z_{[y_{q'}\sigma_0]=E} \cdot \\ & R' \end{aligned}$$

Then we have:

Theorem 4: Assume $\varphi_0 \not\vdash t$ for any $t \in \{x_1\sigma_0, \dots, x_{p'}\sigma_0, y_1\sigma_0, \dots, y_{q'}\sigma_0\}$ and that $x_i\sigma_0 \neq_{E_{\text{enc}} \cup E_a \cup E_b} y_j\sigma_0$ for all $x_i, y_j \in \text{dom}(\sigma_0)$. Then there exist $S', \varphi', \sigma', r'_1, \dots, r'_k, \S'$ such that

$$(R'', \emptyset, \emptyset) \xrightarrow{r'_1 \dots r'_k \cdot \S'} (S', \varphi', \sigma') \quad (8)$$

is a run in the equational theory E_{enc} and such that:

- 1) if $\varphi \vdash t$ for some $t \in \{x_i\sigma, y_j\sigma \mid x_i, y_j \in \text{dom}(\sigma)\}$, then $\varphi' \vdash_{E_{\text{enc}}} s$ for some $s \in \{x_i^a\sigma', x_i^b\sigma', y_j^a\sigma', y_j^b\sigma' \mid x_i, y_j \in \text{dom}(\sigma)\}$
- 2) if there exist $x_i, y_j \in \text{dom}(\sigma)$ such that $x_i\sigma =_{E_{\text{enc}}} y_j\sigma$ then $x_i^c\sigma' =_{E_{\text{enc}}} y_j^c\sigma'$ for some $c \in \{a, b\}$
- 3) otherwise, if $\varphi \vdash x\sigma$ for some variable $x \in \text{vars}(P) \cap \text{dom}(\sigma)$ (resp. $\text{vars}(Q) \cap \text{dom}(\sigma)$), then $\varphi' \vdash_{E_{\text{enc}}} x^a\sigma'$ (resp. $\varphi' \vdash_{E_{\text{enc}}} x^b\sigma'$).

In this tagged setting, the above theorem intuitively states that any trace on the tagged composition of two protocols can be transformed into a trace of the un-tagged composition, but where the two protocols no longer share secrets.

Example 7: We illustrate the above theorem with an example where the untagged composition of two protocols is not secure. However, using the theorem, we can conclude that the tagged composition is secure.

We consider the processes

$$P_1 = \nu x \cdot \nu y \cdot \mathbf{out}(\{x\}_y)$$

and

$$Q_2 = \nu z \cdot \mathbf{out}(\{z\}_x) \cdot \mathbf{in}(z') \cdot \mathbf{out}(\text{dec}(z', y)) \cdot \mathbf{0}$$

previously defined in Section III-A. We have seen that $\nu x' \cdot \nu y' \cdot (x := x') \cdot (y := y') \cdot Q_2$ preserves the secrecy of z , while the sequential composition $P_1 \cdot Q_2$ (where P_1 is used to create the keys x and y) does not preserve the secrecy of z .

However, the sequential composition $[[P_1^a]] \cdot [[Q_2^b]]$ does preserve the secrecy of z by Theorem 4.

We can also use Theorem 4 to prove tagged variants of Theorem 2 and Theorem 3:

Theorem 5 (Tagged version of Theorem 2): Let $P = \nu k_1 \cdot \dots \cdot \nu k_n \cdot (P_1 \mid P_2)$ and $Q = \nu k \cdot (x_k := k \cdot Q_1 \mid y_k := k \cdot Q_2)$ be processes over $\mathcal{F}_{\text{enc}, h}$ such that:

- 1) P_1 binds x_k and P_2 binds y_k
- 2) $f\nu(P) = \emptyset, f\nu(Q) = \emptyset$ and $\text{vars}(P) \cap \text{vars}(Q) = \{x_k, y_k\}$
- 3) $P \models_{E_{\text{enc}}} \text{Secret}(x_k)$ and $P \models_{E_{\text{enc}}} \text{Secret}(y_k)$
- 4) $Q \models_{E_{\text{enc}}} \text{Secret}(x_k)$ and $Q \models_{E_{\text{enc}}} \text{Secret}(y_k)$ and $Q \models_{E_{\text{enc}}} \text{Secret}(x_s)$

Then $W = \nu k_1 \cdot \dots \cdot \nu k_n \cdot ([[P_1^a]] \cdot [[Q_1^b]] \mid [[P_2^a]] \cdot [[Q_2^b]]) \models_{E_{\text{enc}} \cup E_a \cup E_b} \text{Secret}(x_s)$.

Theorem 6 (Tagged version of Theorem 3): Let $P = \nu k_1 \cdot \dots \cdot \nu k_n \cdot (P_1 \mid P_2)$ be a process over $\mathcal{F}_{\text{enc}, h}$ and let $Q = !(\nu k \cdot (x_k := k \cdot Q_1 \mid y_k := k \cdot Q_2))$ be a process over $\mathcal{F}_{\text{enc}, h}$ such that:

- 1) P_1 binds x_k and P_2 binds y_k
- 2) $f\nu(P) = f\nu(Q) = \emptyset$
- 3) $P \models_{E_a} \text{Secret}(x_k)$ and $P \models_{E_a} \text{Secret}(y_k)$
- 4) $Q \models_{E_b} \text{Secret}(x_k)$ and $Q \models_{E_b} \text{Secret}(y_k)$ and $Q \models_{E_b} \text{Secret}(x_s)$

Then $R = \nu k_1 \cdot \dots \cdot \nu k_n \cdot ([[P_1^a]] \cdot [[Q_1^b]] \mid [[P_2^a]] \cdot [[Q_2^b]]) \models_{E_{\text{enc}} \cup E_a \cup E_b} \text{Secret}(x_s)$.

Theorems 5 and 6 allow us to securely compose key-exchange protocols which make use of symmetric encryption with protocols which use the exchanged keys, as long as the two protocols are tagged differently and if they obey the security requirements detailed above.

We have seen that Example 7 explains the need to tag the encryptions in order to obtain secure composition. One might think that tagging encryptions is sufficient to ensure the security of the composition and that it is not necessary to tag the hash function as well. Unfortunately, this is not true. We end this section on tagging by an example which illustrates why tagging is necessary for the hash function as well.

Example 8: We consider the processes

$$P = \nu x \cdot \mathbf{out}(h(x))$$

and

$$Q = \nu z \cdot \mathbf{in}(y) \cdot [y = h(x)] \cdot \mathbf{out}(z).$$

We have that the protocol P does not reveal x . The protocol $\nu x \cdot Q$ reveals neither z nor x . However, if P is used to instantiate the variable x for Q , we have that

$$P \cdot Q \not\models \text{Secret}(z).$$

By Theorem 4, we have however that the tagged composition does satisfy the secret of z :

$$[[P^a]] \cdot [[Q^b]] \models \text{Secret}(z).$$

VI. RELATED WORK

There are two large classes of models for studying the security of cryptographic protocols. On one hand, there are the Dolev-Yao (also called symbolic) models, in which messages sent over the network are represented by terms and the attacker is modeled as a deduction system. On the other hand, there are the computational (or cryptographic) models, in which the messages are bit-strings and the attacker is an arbitrary probabilistic polynomial time Turing machine.

Our result clearly belongs to the first approach. One of the first papers studying the composition of protocols in the symbolic model is [11]. In this paper, Guttman and Thayer show that (in the formalism of strand spaces [26]) two protocols which make use of concatenation and encryption can be safely executed together without damaging interactions, as soon as the protocols are “independent”. Also, an assumption is made that all keys are atomic and not generated for example by hashing some message. The independence hypothesis requires in particular that the sets of encrypted messages handled by the two protocols be disjoint. This is a semantic hypothesis on all possible executions of the two protocols which needs to be checked by hand.

In [13], Cortier *et al* show that tagging is sufficient to avoid collusion between protocols sharing common keys and making use of standard cryptographic primitives: concatenation, signature, hash functions and encryption. This framework allows to compose processes in parallel; however, it does not allow to securely compose e.g. a key exchange protocol with another protocol which makes use of the shared key. In particular, this is because the shared keys should never appear as payloads.

In [12], Guttman provides a characterization which ensures that two protocols can run securely together when sharing some data such as keys or payloads. The main improvement over [11] is that keys are allowed to be non-atomic. The characterization is syntactic but has to be computed for each pair of protocols. As cryptographic primitives, the protocols are allowed to contain encryptions and concatenations. The proof method in our result is roughly similar to the proof methods described here: an attack against the composition is transformed into an attack against one of the two protocols.

In [14], Delaune *et al* use a derivative of the applied- π calculus to model off-line guessing attacks. They show that in the passive case resistance against guessing attacks is preserved by the composition of two protocols which share the weak secret against which the attack is mounted. This result (in the passive case) holds for arbitrary equational theories. However, for the active case this is no longer the case: it is however proven that *tagging* the weak secret enforces secure composition (in the sense of guessing attacks). Again, this framework applies to parallel composition only.

Mödersheim and Viganò [15] have proposed a framework for composing protocols sequentially. They propose a criterion for a protocol P_1 to safely use P_2 as a sub-protocol (for implementing a secure channel). However, their criterion is a semantic criterion, for which no decision procedure has been provided yet.

In [27], Delaune *et al* use a simulation based approach inspired from the computational model to provide a framework

for securely composing protocols in the applied- π calculus. This involves defining for each sub-protocol an *ideal functionality* and then showing that a certain implementation securely emulates the ideal functionality.

Another line of work is represented by the Protocol Composition Logic [16], which can be used to modularly prove security properties of protocols using a fixed set of primitives. In order to safely compose two protocols, one has to check that each protocol satisfies some invariant used in the security proof of the other protocol. While offering more flexibility, this criteria is not syntactic and needs to be checked each time by hand.

As opposed to [11], [13], [12], [15], [16], our result allows not only the standard cryptographic primitives like encryption and hash functions, but arbitrary primitives expressible as equational theories. Furthermore, unlike [13], [14], our result allows to compose protocols asymmetrically (i.e. not just in parallel). The main difference between our approach and [27] is that we do not need to prove anything about the protocols we are trying to compose except standard reachability properties. In particular, we do not have to provide a key exchange functionality and prove that an implementation satisfies this functionality. However, [27] can be used to reason about protocols which do share primitives.

In the context of computational models, Canetti *et al.* have developed the Universal Composability framework [17], designed to allow composition. In this framework, an *ideal functionality* is defined and a specific protocol is shown to implement this functionality securely. Then this protocol may be securely used instead of the functionality. This approach is compositional in the sense that the protocol can be safely used instead of the functionality in any context (possibly inside other protocols/functions). However, as pointed out in [18], this framework does not allow *a priori* to compose protocols *sharing* data such as keys. Some specific results have been further developed in order to allow composition with “joint state” [18]. These results allow e.g. several sessions of a protocol (sharing common data such as keys and random coins) to be considered independently (each session having fresh keys and randomness). However, the shared data have to be used in the same manner in each copies. It is not possible for example to use this approach for composing a protocol that uses a key for encrypting data with a protocol that uses the same key as payload (even if the key remains secret), as it is done in our work.

VII. CONCLUSIONS AND FUTURE WORK

We have proven that protocols can be securely composed provided that they use primitives modeled by disjoint equational theories or provided that their common primitives are tagged encryptions or tagged hash functions.

Our result is a generic composition result: any trace leading to an attack on the composition of the two protocols is transformed into a trace that leads to an attack in one of the individual protocols, even if the two protocols share secrets such as keys. This allows us to securely perform several kinds of compositions. We can have secure parallel composition under

shared secrets or we can have an asymmetric composition, where one of the protocols is used as a sub-protocol. As a matter of fact, our combination theorem could actually be used in any context where two protocols are arbitrarily interleaved and use shared data.

As an application, we have shown how our main composition theorem can be used in order to securely refine a protocol that uses pre-established keys or secure channels.

For the sake of simplicity, the only security property that we have considered is secrecy. We believe however that our result extends to general trace properties (e.g. authentication). This is because our trace transformation proof technique transforms any trace of the composition of two protocols under shared secrets (as long as a shared key is not revealed) into a trace on the composition under no shared secrets. This means that any violation of authentication in the composed protocol would be transformed into a violation of authentication on one of the individual protocols.

We intend to develop and analyze a logic for trace properties which are preserved by our composition theorem. We are also investigating if composition with disjoint equational theories preserves *trace equivalence*, as defined e.g. in [28] and more generally other behavioral equivalences which can be used to reason about security properties such as anonymity.

We have proven that primitives can be shared between the protocols provided they are tagged, in the case of symmetric encryption and hash. We think that our proof technique easily extends to any classical destructor/constructor theories (e.g. signatures and asymmetric encryption).

There are certain primitives which seem harmless enough that they may be shared without tagging them. For example, the concatenation defined through the equational theory:

$$fst(pair(x, y)) = x \quad snd(pair(x, y)) = y.$$

is a candidate. However, let us consider the processes

$$P = \nu x \cdot \nu y \cdot z := pair(x, y)$$

$$Q = \nu k \cdot \mathbf{out}(enc(z, k)) \cdot \mathbf{in}(y) \cdot \mathbf{out}(pair(fst(dec(y, k)), snd(dec(y, k)))).$$

The process $\nu z \cdot Q$ does not reveal z . However, if the generation of z is handled by P , we have that $P \cdot Q$ does reveal z . This is because Q was only verified secure when z is instantiated to a name. We are trying to prove that the equational theory of *pair* can be safely shared between two protocols as long as neither of the protocols instantiates a shared key to a pair.

Finally, many relevant equational theories are not so easy to tag. In particular, tagging *exclusive or* is particularly difficult. Finding a way to securely compose two protocols which both make use of this primitive (the *exclusive or*) is a challenging open problem.

REFERENCES

- [1] G. Lowe, "Breaking and fixing the Needham-Schroeder public-key protocol using FDR," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, ser. LNCS, T. Margaria and B. Steffen, Eds., vol. 1055. Springer-Verlag, march 1996, pp. 147–166.
- [2] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hanks Drielsma, P.-C. 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 applications," in *17th International Conference on Computer Aided Verification, CAV'2005*, ser. Lecture Notes in Computer Science, K. Etessami and S. Rajamani, Eds., vol. 3576. Edinburgh, Scotland: Springer, 2005, pp. 281–285.
- [3] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. T. Abad, "Formal analysis of saml 2.0 web browser single sign-on: Breaking the saml-based single sign-on for google apps," in *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*, 2008, pp. 1–10.
- [4] M. Abadi and B. Blanchet, "Computer-Assisted Verification of a Protocol for Certified Email," *Science of Computer Programming*, vol. 58, no. 1–2, pp. 3–27, Oct. 2005, special issue SAS'03.
- [5] M. Abadi, B. Blanchet, and C. Fournet, "Just Fast Keying in the Pi Calculus," in *Programming Languages and Systems: Proceedings of the 13th European Symposium on Programming (ESOP'04)*, ser. Lecture Notes on Computer Science, D. Schmidt, Ed., vol. 2986. Barcelona, Spain: Springer Verlag, Mar. 2004, pp. 340–354.
- [6] 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.
- [7] M. Rusinowitch and M. Turuani, "Protocol Insecurity with Finite Number of Sessions and Composed Keys is NP-complete," *Theoretical Computer Science*, vol. 299, pp. 451–475, April 2003. [Online]. Available: <http://www.loria.fr/rusi/pub/tcsprotocol.ps.gz>
- [8] B. Blanchet, "An efficient cryptographic protocol verifier based on prolog rules," in *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press, June 2001.
- [9] G. Lowe, "Casper: A compiler for the analysis of security protocols," in *Proc. of 10th Computer Security Foundations Workshop (CSFW'97)*. Rockport, Massachusetts, USA: IEEE Computer Society Press, 1997, also in *Journal of Computer Security*, Volume 6, pages 53–84, 1998.
- [10] B. Blanchet, "An automatic security protocol verifier based on resolution theorem proving (invited tutorial)," in *20th International Conference on Automated Deduction (CADE-20)*, Tallinn, Estonia, July 2005.
- [11] J. D. Guttman and F. J. Thayer, "Protocol independence through disjoint encryption," in *Proc. 13th Computer Security Foundations Workshop (CSFW'00)*. IEEE Comp. Soc. Press, 2000, pp. 24–34.
- [12] J. D. Guttman, "Cryptographic protocol composition via the authentication tests," in *Foundations of Software Science and Computation Structures (FOSSACS'09)*, ser. Lecture Notes in Computer Science, March 2009.
- [13] V. Cortier, J. Delaitre, and S. Delaune, "Safely composing security protocols," in *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, ser. Lecture Notes in Computer Science, V. Arvind and S. Prasad, Eds., vol. 4855. New Delhi, India: Springer, Dec. 2007, pp. 352–363. [Online]. Available: <http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/CDD-fsttcs07.pdf>
- [14] S. Delaune, S. Kremer, and M. D. Ryan, "Composition of password-based protocols," in *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*. Pittsburgh, PA, USA: IEEE Computer Society Press, Jun. 2008, pp. 239–251. [Online]. Available: <http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/DKR-csf08.pdf>
- [15] S. Mödersheim and L. Viganò, "Secure pseudonymous channels," in *Proceedings of the 14th European Symposium On Research In Computer Security (ESORICS'09)*, ser. Lecture Notes in Computer Science, vol. 5789. Springer, 2009, pp. 337–354.
- [16] A. Datta, A. Derek, J. C. Mitchell, and A. Roy, "Protocol composition logic (pcl)," *Electron. Notes Theor. Comput. Sci.*, vol. 172, pp. 311–358, 2007.
- [17] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *IEEE Symposium on Foundations of Computer Science*, 2001, pp. 136–145. [Online]. Available: citeseer.ist.psu.edu/article/canetti05universally.html
- [18] R. Canetti and T. Rabin, "Universal composition with joint state," in *CRYPTO*, 2003, pp. 265–281.
- [19] M. Backes, M. Maffei, and D. Unruh, "Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol," in *IEEE Symposium on Security and Privacy*, 2008, pp. 202–215.
- [20] S. Delaune, S. Kremer, and M. D. Ryan, "Verifying privacy-type properties of electronic voting protocols," *Journal of Computer*

- Security*, vol. 17, no. 4, pp. 435–487, Jul. 2009. [Online]. Available: <http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/DKR-jcs08.pdf>
- [21] M. Abadi and C. Fournet, “Mobile values, new names, and secure communication,” in *Proc. of the 28th ACM Symposium on Principles of Programming Languages (POPL’01)*, January 2001, pp. 104–115.
- [22] F. Baader and T. Nipkow, *Term Rewriting and All That*. Cambridge University Press, 1999.
- [23] B. Blanchet and A. Podelski, “Verification of cryptographic protocols: Tagging enforces termination,” in *Foundations of Software Science and Computation Structures (FoSSaCS’03)*, ser. LNCS, A. Gordon, Ed., vol. 2620, April 2003.
- [24] R. Ramanujam and S.P.Suresh, “Tagging makes secrecy decidable for unbounded nonces as well,” in *Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS’03)*, Mumbai, 2003.
- [25] J. Heather, G. Lowe, and S. Schneider, “How to prevent type flaw attacks on security protocols,” in *Proc. of the 13th Computer Security Foundations Workshop (CSFW’00)*. IEEE Computer Society Press, 2000.
- [26] F. J. T. Fábrega, “Strand spaces: proving security protocols correct,” *J. Comput. Secur.*, vol. 7, no. 2-3, pp. 191–230, 1999.
- [27] S. Delaune, S. Kremer, and O. Pereira, “Simulation based security in the applied pi calculus,” in *Proceedings of the 29th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS’09)*, ser. Leibniz International Proceedings in Informatics, R. Kannan and K. Narayan Kumar, Eds., vol. 4. Kanpur, India: Leibniz-Zentrum für Informatik, Dec. 2009, pp. 169–180. [Online]. Available: <http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/DKP-fsttcs09.pdf>
- [28] V. Cortier and S. Delaune, “A method for proving observational equivalence,” in *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF’09)*. Port Jefferson, NY, USA: IEEE Computer Society Press, Jul. 2009, pp. 266–276. [Online]. Available: <http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/CD-csf09.pdf>

APPENDIX

A. Combination of equational theories

Proposition 1: All alien subterms of a collapsed term are collapsed.

Proof:

First note that $|\text{collapse}(t)| \leq |t|$. Indeed, by induction on t , we have either:

$$|\text{collapse}(t)| = |s_j| \leq |\text{collapse}(t_x)| \leq |t_x| < |t|$$

for some $x \in \{1, \dots, n\}$ (since s_j is a subterm of t_x) or

$$|\text{collapse}(t)| = |C[\text{collapse}(t_1), \dots, \text{collapse}(t_n)]| \leq |C[t_1, \dots, t_n]|$$

Let t be a term. We prove by induction on $|t|$ that all alien subterms of $\text{collapse}(t)$ are collapsed. Let $t = C[[t_1, \dots, t_n]]$ and $C[\text{collapse}(t_1), \dots, \text{collapse}(t_n)] = D[[s_1, \dots, s_k]]$. We distinguish two cases.

- 1) $D[n_{[s_1]=E}, \dots, n_{[s_k]=E}] =E n_{[s_j]=E}$ for some j , in which case $\text{collapse}(t) = s_j$.

But s_j is either $\text{collapse}(t_x)$ or an alien subterm of $\text{collapse}(t_x)$ for some $x \in \{1, \dots, n\}$. In the first case, we have that $s_j = \text{collapse}(t_x)$ is collapsed and we can apply the induction hypothesis on $s_j = \text{collapse}(t_x)$. In the second case, s_j is an alien subterm of $\text{collapse}(t_x)$ and by the induction hypothesis is collapsed. Therefore we can apply the induction hypothesis on s_j itself to conclude.

- 2) $\text{collapse}(t) = C[\text{collapse}(t_1), \dots, \text{collapse}(t_n)]$, in which case any alien subterm of $\text{collapse}(t)$ is
 - a) either $\text{collapse}(t_x)$ for some $x \in \{1, \dots, n\}$ (in which case we are done, as the alien term $\text{collapse}(t_x)$ is collapsed)
 - b) or an alien subterm of $\text{collapse}(t_x)$, in which case we conclude by the induction hypothesis on t_x

■

B. Proof of the main result

Claim 1: If t_1, t_2 are collapsed terms such that $t_1 =E t_2$, then $V_c(t_1) =E V_c(t_2)$ ($c \in \{a, b\}$).

Proof:

By induction on the size of terms. By Lemma 3, t_1 and t_2 start with a root symbol from the same signature. Assume w.l.o.g. that $\text{root}(t_1), \text{root}(t_2) \in \mathcal{F}_a$.

If $t_1 =E t_2 =E s$ for some $s \in \{x_1\sigma_0, \dots, x_{p'}\sigma_0, y_1\sigma_0, \dots, y_{q'}\sigma_0\}$ and $c = b$, then $V_b(t_1) = n_{[t_1]=E} = n_{[t_2]=E} = V_b(t_2)$ and we are done.

Otherwise, $t_1 = C[[s_1, \dots, s_n]]$, $t_2 = D[[s'_1, \dots, s'_m]]$ for some pure a -contexts C and D . Then $V_c(t_1) = C[V_a(s_1), \dots, V_a(s_n)]$ and $V_c(t_2) = D[V_a(s'_1), \dots, V_a(s'_m)]$ for any $c \in \{a, b\}$.

We can apply the induction hypothesis on $s_1, \dots, s_n, s'_1, \dots, s'_m$ (which are collapsed by Proposition 1) to conclude that the equalities between these terms are preserved when passed through $V_c(_)$ and therefore we can apply Lemma 1 to conclude.

■

Claim 2: We have that

$$V_c(r\varphi) =_{\text{E}} V_c(\text{collapse}(r\varphi))$$

and that $\text{collapse}(r\varphi)$ is a good term.

Proof: Indeed, let C be such that $r\varphi = C[[t_1, \dots, t_k]]$. As $\varphi = \{w_j \mapsto s_j\sigma\}_j$ for some pure terms s_j , C either fully spans s_j or it does not span it at all (for all j). Therefore each term t_i ($1 \leq i \leq k$) is such that t_i is a subterm of σ or $t_i = r_i\varphi$ for some recipe r_i (where r_i is a subterm of r).

Let the predicate $P(i)$ be true exactly on indexes i such that $t_i = r_i\varphi$ (where r_i is a subterm of r).

By the induction hypothesis, for all indexes i such that $P(i)$, we have that $V_c(r_i\varphi) =_{\text{E}} V_c(\text{collapse}(r_i\varphi))$ and $\text{collapse}(r_i\varphi) = E_i[[s_1^i, \dots, s_{n_i}^i]]$ is a good term. Also $s_1^i, \dots, s_{n_i}^i$ must be good terms ($1 \leq i \leq k$)¹. We have thus defined E_i and s_j^i for all i such that $P(i)$.

For all indexes i such that not $P(i)$, t_i is an alien subterm of σ (and is therefore collapsed) and therefore we have that $t_i = E_i[[s_1^i, \dots, s_{n_i}^i]]$ is a good term (by the first item of the definition) and that s_j^i ($1 \leq i \leq k, 1 \leq j \leq n_i$) are good terms (by the first items of the definition as well). We have thus defined E_i and s_j^i for all i such that not $P(i)$.

We next define a class of contexts C_i ($1 \leq i \leq k$). Let $C_i = E_i$ if $\text{root}(E_i)$ comes from the same signature as $\text{root}(C)$ and let $C_i = _$ otherwise.

Let $C' = C[C_1, \dots, C_k]$. Then

$$\begin{aligned} r\varphi &= C[[t_1, \dots, t_k]] \\ &=_{\text{E}} C[E_1[[s_1^1, \dots, s_{n_1}^1]], \dots, E_k[[s_1^k, \dots, s_{n_k}^k]]] \\ &= C'[[s_1, \dots, s_l]] \end{aligned}$$

for good terms s_j ($1 \leq j \leq l$)² and furthermore, by the definition of collapse

$$\text{collapse}(r\varphi) = s_j \quad (9)$$

if $C'[n_{[s_1]=\text{E}}, \dots, n_{[s_l]=\text{E}}] =_{\text{E}} n_{[s_j]=\text{E}}$ for some j and fresh names $n_{[_]=\text{E}}$ or

$$\text{collapse}(r\varphi) = C'[[s_1, \dots, s_l]] \quad (10)$$

otherwise. In either case, we can see that $\text{collapse}(r\varphi)$ is a good term³, which concludes part of the induction.

Let us now prove that

$$V_c(r\varphi) =_{\text{E}} C'[V_d(s_1), \dots, V_d(s_l)] \quad (11)$$

where d is such that $\text{root}(C) \in \mathcal{F}_d$

Indeed, by the definition of V_c and because $r\varphi$ cannot be both a shared secret in $\{x_1, \dots, x_{p'}, y_1, \dots, y_{q'}\}\sigma_0$ and have its root symbol in \mathcal{F}_c (by hypothesis), we have that

$$V_c(r\varphi) = C[V_d(t_1), \dots, V_d(t_k)] \quad (12)$$

¹if $\text{collapse}(r_i\varphi)$ is good by the first item of the definition of good, then s_j^i are subterms of it and therefore subterms of σ , which means they are good; if $\text{collapse}(r_i\varphi)$ is good by the second item of the definition of good, then s_j^i are good by definition

²a term s_j is either equal to some t_j if $C_j = _$ and we know t_j s are good, or is equal to some s_x^i , which we also know are good

³because s_j is a good term by the first item in the definition and because $C'[[s_1, \dots, s_l]]$ is a good term by the second item of the definition

We prove that

$$V_d(t_i) =_{\text{E}} V_d(\text{collapse}(t_i)) \quad (13)$$

Indeed, either t_i is an alien subterm of σ , in which case $t_i = \text{collapse}(t_i)$ and Equation 13 trivially holds, or $t_i = r_i\varphi$, in which case Equation 13 holds by the induction hypothesis.

From Equations 12 and 13, as $\text{collapse}(t_i) = E_i[[s_1^i, \dots, s_{n_i}^i]]$, we have that

$$V_c(r\varphi) = C[V_d(E_1[[s_1^1, \dots, s_{n_1}^1]]), \dots, V_d(E_k[[s_1^k, \dots, s_{n_k}^k]])] \quad (14)$$

Let us prove that for each i such that $C_i = E_i$ that

$$V_c(E_i[s_1^i, \dots, s_{n_i}^i]) = E_i[V_d(s_1^i), \dots, V_d(s_{n_i}^i)] \quad (15)$$

Indeed, for such an i , t_i cannot be a shared secret in $\{x_1, \dots, x_{p'}, y_1, \dots, y_{q'}\}\sigma_0$ (by hypothesis, as $t_i =_{\text{E}} r_i\varphi$ for some strict subrecipe r' of r) and therefore Equation 15 holds by the definition of collapse and because $\text{root}(E_i), \text{root}(C) \in \mathcal{F}_d$ ($\text{root}(E_i)$ and $\text{root}(C)$ are from the same signature).

From Equations 15 and 14, and by the definition of $C' = C[C_1, \dots, C_k]$, we immediately obtain Equation 11.

We are now ready to prove that

$$V_c(r\varphi) =_{\text{E}} V_c(\text{collapse}(r\varphi)) \quad (16)$$

We distinguish between two cases (either Equation 9 or Equation 10 holds).

- if Equation 9 holds then we must have that

$$C'[n_{[s_1]=\text{E}}, \dots, n_{[s_l]=\text{E}}] =_{\text{E}} n_{[s_j]=\text{E}}$$

and that

$$V_c(\text{collapse}(r\varphi)) = V_c(s_j) \quad (17)$$

But as s_1, \dots, s_l are good terms, they are also collapsed and by Claim 1, we have that $s_x =_{\text{E}} s_y$ implies $V_d(s_x) =_{\text{E}} V_d(s_y)$ ($1 \leq x, y \leq l$). Therefore

$$C'[V_d(s_1), \dots, V_d(s_l)] =_{\text{E}} V_d(s_j)$$

Combining this with Equation 11, we obtain

$$V_c(r\varphi) =_{\text{E}} V_d(s_j) \quad (18)$$

- if $s_j = \text{collapse}(r\varphi) =_{\text{E}} r\varphi$ is not a shared secret in $\{x_1, \dots, x_{p'}, y_1, \dots, y_{q'}\}\sigma_0$ then

$$V_c(s_j) = V_d(s_j)$$

which combined with Equations 17 and 18 immediately yields Equation 16.

- if $s_j =_{\text{E}} r\varphi$ is a shared secret in $\{x_1, \dots, x_{p'}, y_1, \dots, y_{q'}\}$, then by hypothesis $\text{root}(r\varphi) \in \mathcal{F}_c$. But \mathcal{F}_d was chosen as the signature which contains the root symbol of C (equivalently the signature which contains the root symbol of $r\varphi$) and therefore $d = c$ which combined with Equations 17 and 18 immediately yields Equation 16.

- if Equation 10 holds then

$$V_c(\text{collapse}(r\varphi)) = C'[[V_d(s_1), \dots, V_d(s_l)]]$$

by the definition of *collapse*. Combining this with Equation 11, we immediately obtain Equation 16. \blacksquare

Claim 3: For $i \in \{0, \dots, n\}$, we have that $\text{collapse}(w_j \varphi_i)$ (for all $w_j \in \text{dom}(\varphi_i)$) is an i -good term and $x\sigma_i = C[s_1, \dots, s_k]$ (for all $x \in \text{dom}(\sigma_i)$) for a pure (possibly empty) \mathcal{F}_c -context C and $i-1$ -good terms s_1, \dots, s_k (with $\text{root}(s_j) \in \mathcal{F}_c$), where $c = a$ if $x \in \text{vars}(P) \setminus \{x_1, \dots, x_{p'}\}$ and $c = b$ if $x \in \text{vars}(Q) \setminus \{y_1, \dots, y_{q'}\}$.

Proof: By induction on i . The base case $i = 0$ is particularly trivial, since $\text{dom}(\sigma_0) = \emptyset$ and $\text{dom}(\varphi_0) = \emptyset$. Assuming the hypothesis holds for $j \leq i-1$, we prove it for i . We do a case analysis on the transition \S_i :

- 1) if the i -th step was the execution of some νx , then $\varphi_i = \varphi_{i-1}$ and $\sigma_i = \sigma_{i-1} \cup \{x \mapsto m\}$ where m is a fresh name from \mathcal{F}_c , where c is defined as in the lemma. Therefore it is sufficient to chose $C = m$ and $k = 0$ to conclude.
- 2) if the i -th step was an assignment $x := t$, where t is a pure \mathcal{F}_c term and x, c are as in the lemma, we have that $x\sigma_i = \text{collapse}(t\sigma_{i-1})$. But $t\sigma_{i-1} = C[s_1, \dots, s_k]$ for a pure \mathcal{F}_c -context C and terms $s_j = y_j \sigma_{i-1}$ (C is obtained from t by replacing variables with holes). By the induction hypothesis, $s_j = C_j[s_1^j, \dots, s_{n_j}^j]$ for $i-2$ -good terms s_x^j and a pure context C_j . Let C'_j be $_$ if $C_j = _$ or if $\text{root}(C_j) \in \mathcal{F}_c$ and let $C'_j = C_j$ otherwise. Let $C' = C[C'_1, \dots, C'_k]$. Then $t\sigma_{i-1} = C'[[t_1, \dots, t_l]]$ (each t_j is either some $s_{j'}$ or some s_x^j). Now $\text{collapse}(t\sigma_{i-1})$ is equal either to some t_j (which are $i-2$ -good, in which case we conclude, or $\text{collapse}(t\sigma_{i-1}) = C'[[t_1, \dots, t_l]]$, in which case we are done (we have identified the C from the lemma statement (it is C') and the s_1, \dots, s_k from the lemma statement (they are t_1, \dots, t_l)).
- 3) if the i -th statement is an output $\text{out}(t)$, then we conclude that $t\sigma_i$ is an $i-1$ -good term exactly as above (for assignment) (except we need that $C'[[t_1, \dots, t_l]] = t\sigma_{i-1}$ be deducible from φ_i , which is the case (it has just been output)).
- 4) if the i -th step was an input $\text{in}(x)$, then there exists a context C and i -good terms t_1, \dots, t_k such that $x\sigma_i = \text{collapse}(C[t_1, \dots, t_k])$ (we define C to be the recipe \S_i in which the variables have been replaced by holes and t_1, \dots, t_k are the corresponding elements from φ_{i-1} , which we know by induction to be $i-2$ -good terms). We prove by structural induction on C that $\text{collapse}(C[t_1, \dots, t_k])$ is an $i-1$ -good term. Indeed, let C' be such that $C[t_1, \dots, t_k] = C'[[s_1, \dots, s_l]]$. Then each s_j is either equal to $C_j[t_1, \dots, t_k]$ (for some C_j subcontext of C) or s_j is an i -good term (because it is an alien subterm of some i -good term $t_{j'}$ which cannot be a shared key in $\{x_1, \dots, x_{p'}, y_1, \dots, y_{q'}\}\sigma_i$). In either case, we know (either by the induction hypothesis in the first case or by definition of i -good in the second case) that $\text{collapse}(s_j)$ is a i -good term and therefore $\text{collapse}(s_j) = C_j[[s_1^j, \dots, s_{n_j}^j]]$ for some context C_j and some i -good terms s_x^j or s_j

is a shared secret in $\{x_1, \dots, x_{p'}, y_1, \dots, y_{q'}\}\sigma$. We let $C'_j = C_j$ if s_j is not a shared secret and if $\text{root}(C'_j)$ is from the same signature as C and $C'_j = _$ otherwise. Let $C'' = C'[C'_1, \dots, C'_k]$. Then we have that $C[t_1, \dots, t_k] =_{\text{E}} C''[u_1, \dots, u_m]$ for some i -good terms u_j (each u_j is either some $\text{collapse}(s_{j'})$ or some s_x^j) and that $\text{collapse}(C[t_1, \dots, t_k])$ is either $C''[u_1, \dots, u_m]$ or some u_j . In either case, we can immediately conclude. \blacksquare

Lemma 6: If P is a linear process over \mathcal{F}_a and

$$(P, \emptyset, \emptyset) \xrightarrow{\S_1}_{\text{E}} (P_1, \varphi_1, \sigma_1) \xrightarrow{\S_2}_{\text{E}} \dots \xrightarrow{\S_n}_{\text{E}} (P_n, \varphi_n, \sigma_n)$$

then

$$(P, \emptyset, \emptyset) \xrightarrow{\S'_1}_{\text{E}_a} (P_1, \varphi'_1, \sigma'_1) \xrightarrow{\S'_2}_{\text{E}_a} \dots \xrightarrow{\S'_n}_{\text{E}_a} (P_n, \varphi'_n, \sigma'_n)$$

for some φ'_n, σ'_n and \S'_n such that $\varphi_n \vdash_{\text{E}} x\sigma_n$ implies $\varphi'_n \vdash_{\text{E}_a} x\sigma_n$ and $x\sigma_n =_{\text{E}} y\sigma_n$ implies $x\sigma'_n =_{\text{E}_a} y\sigma'_n$ for all $x, y \in \text{dom}(\sigma_n)$.

Proof:

Let $\text{init}/0$ be any constant in \mathcal{F}_a (we have assumed that there exists at least such a constant). The idea is that we obtain the second trace from the first trace by abstracting any elements that starts with a symbol from \mathcal{F}_b by the constant init .

This abstraction is formalized by the function *abstract*, which is defined inductively on ground terms as follows:

- 1) $\text{abstract}(f(t_1, \dots, t_k)) = f(\text{abstract}(t_1), \dots, \text{abstract}(t_k))$ if $f \in \mathcal{F}_a$
- 2) $\text{abstract}(f(t_1, \dots, t_k)) = \text{init}$ if $f \in \mathcal{F}_b$

The function *abstract* enjoys the following good property on collapsed terms:

Lemma 9: If $s =_{\text{E}} t$ and s and t are collapsed, we have that $\text{abstract}(s) =_{\text{E}_a} \text{abstract}(t)$.

Proof: Let $s = C[[s_1, \dots, s_k]]$ and $t = D[[t_1, \dots, t_l]]$. As $s =_{\text{E}} t$ and s and t are collapsed, by Lemma 1, we have that $\text{root}(C)$ and $\text{root}(D)$ come from the same signature \mathcal{F}_c . If $c = b$, then $\text{abstract}(s) = \text{abstract}(t) = \text{init}$ and we are done.

Otherwise $c = a$ and $\text{abstract}(s) = C[\text{init}, \dots, \text{init}]$ and $\text{abstract}(t) = D[\text{init}, \dots, \text{init}]$. By Lemma 1, we have that $C[n_{[s_1]=\text{E}}, \dots, n_{[s_k]=\text{E}}] =_{\text{E}_a} D[n_{[t_1]=\text{E}}, \dots, n_{[t_l]=\text{E}}]$, where $n_{_}$ are fresh names. As our theory is stable by replacement of arbitrary terms for names, we have that $(C[n_{[s_1]=\text{E}}, \dots, n_{[s_k]=\text{E}}] =_{\text{E}_a} D[n_{[t_1]=\text{E}}, \dots, n_{[t_l]=\text{E}}]) \{n_{_} \mapsto \text{init}\}$ and therefore $\text{abstract}(s) =_{\text{E}_a} \text{abstract}(t)$. \blacksquare

We also have that:

Lemma 10: If t is a pure \mathcal{F}_a term and σ is a collapsed substitution, we have that $\text{abstract}(\text{collapse}(t\sigma)) =_{\text{E}_a} \text{abstract}(t\sigma)$.

Proof:

Indeed, if t is a variable then $\text{collapse}(t\sigma) = t\sigma$ (since we assumed σ to be collapsed) and we are done.

Otherwise, $\text{root}(t) \in \mathcal{F}_a$. Let C be the context obtained from t by replacing all variables with holes. Then $t\sigma = C[t_1, \dots, t_k]$ for some collapsed terms t_j ($1 \leq j \leq k$).

Let $t_j = C_j[[t_1^j, \dots, t_{k_j}^j]]$. Let $C'_j = C_j$ if $\text{root}(C_j) \in \mathcal{F}_a$ and let $C'_j = _$ otherwise. Let $C' = C[C'_1, \dots, C'_j]$. Then $t\sigma = C'[[s_1, \dots, s_l]]$.

Then either $\text{collapse}(t\sigma) = t\sigma$ and we are done or $C'[n_{[s_1]=\varepsilon}, \dots, n_{[s_l]=\varepsilon}] =_{\varepsilon} n_j$ and $\text{collapse}(t\sigma) = s_j$ for some $1 \leq j \leq l$. As our equational theory is stable by replacement of terms for names, we have that

$$(C'[n_{[s_1]=\varepsilon}, \dots, n_{[s_l]=\varepsilon}] =_{\varepsilon} n_j)\{n_- \mapsto \text{init}\}.$$

But $\text{abstract}(t\sigma) = C'[\text{init}, \dots, \text{init}]$ and $\text{abstract}(\text{collapse}(t\sigma)) = \text{init}$ and therefore we conclude. \blacksquare

We now define σ'_i and φ'_i (for $1 \leq i \leq n$). We assume w.l.o.g. that σ_i and φ_i are collapsed and let $\sigma'_i(x) = \text{abstract}(\sigma(x))$ and $\varphi'_i(w) = \text{abstract}(\varphi(w))$ ($1 \leq i \leq n$, $x \in \text{dom}(\sigma_i)$, $w \in \text{dom}(\varphi_i)$).

Definition 11: We say that a collapsed term $t = C[[t_1, \dots, t_k]]$ is i -good ($1 \leq i \leq n$) if there exist pure \mathcal{F}_a recipes r_1, \dots, r_k such that $\varphi'_i \vdash_{\varepsilon_a}^{r_j} \text{abstract}(t_j)$ and if t_j is an i -good term ($1 \leq j \leq k$).

It is easy to see that a $(i-1)$ -good term is also an i -good term.

We prove by induction on i that there exist ξ'_1, \dots, ξ'_i such that the recipes among them are pure \mathcal{F}_a -recipes such that:

$$(P, \emptyset, \emptyset) \xrightarrow{\xi'_1} (P_1, \varphi'_1, \sigma'_1) \xrightarrow{\xi'_2} \dots \xrightarrow{\xi'_i} (P_i, \varphi'_i, \sigma'_i)$$

and that $\varphi_i(w)$ and $\sigma_i(x)$ are i -good ($w \in \text{dom}(\varphi_i)$ and $x \in \text{dom}(\sigma_i)$).

We distinguish among the possible actions at step i :

- 1) if the action was a νx , then ξ'_i is the empty string. We choose ξ'_i also as the empty string.

By definition of the transition, we have that $\sigma_i = \sigma_{i-1} \cup \{x \mapsto n\}$ for some fresh name n . We only need to show that $\sigma_i(x)$ is an i -good term, since the other terms are i -good directly by the induction hypothesis. We $C = n$ (a context with 0 holes) and $k = 0$ in the definition of i -good and we can trivially conclude that $\sigma_i(x)$ is i -good.

We also need to establish that this transition works, i.e. $\sigma'_i(x) = \text{abstract}(\sigma_i(x))$, which is obviously the case.

- 2) if the action was an assignment $x := t$, then ξ'_i is the empty string. We choose ξ'_i also as the empty string.

By definition of the transition, we have that $\sigma_i = \sigma_{i-1} \cup \{x \mapsto \text{collapse}(t\sigma_{i-1})\}$. As t is a term appearing in the protocol, it is a pure \mathcal{F}_a -term.

Let C be the context obtained from t by replacing all variables with holes. Then $\sigma_i(x) = C[t_1, \dots, t_k]$ for some i -good terms t_j ($1 \leq j \leq k$) – the terms t_j are equal to $\sigma_{i-1}(y_j)$ for some $y_j \in \text{dom}(\sigma_{i-1})$ ($1 \leq j \leq k$). Let $t_j = C_j[[t_1^j, \dots, t_{k_j}^j]]$.

If C is the empty context we have that $\sigma_i(x) = \sigma_{i-1}(y)$ for some $y \in \text{dom}(\sigma_{i-1})$ and therefore we can conclude by the induction hypothesis.

Otherwise, if $C \neq _$, let c be such that $\text{root}(C) \in \mathcal{F}_c$. For $1 \leq j \leq k$, let $C'_j = C_j$ if $\text{root}(C_j) \in \mathcal{F}_c$ and let $C_j = _$ otherwise. Let $C' = C[C'_1, \dots, C'_k]$. We have

that $t\sigma_{i-1} = C'[[s_1, \dots, s_l]]$ such that s_j is an i -good term (each s_j ($1 \leq j \leq l$) is either some $t_{j'}$ ($1 \leq j' \leq k$) which is i -good by the induction hypothesis or some $t_z^{j'}$ ($1 \leq j' \leq k$, $1 \leq z \leq k_j$) which is i -good because it is an alien term of $t_{j'}$, which is i -good by the induction hypothesis).

Then $\text{collapse}(t\sigma_{i-1})$ is either equal to some s_j ($1 \leq j \leq l$) or to $C[s_1, \dots, s_l]$.

In the first case, we conclude because we have already seen that all s_j ($1 \leq j \leq l$) are i -good.

In the second case, we have that $\text{root}(C) \in \mathcal{F}_a$ and therefore $\text{root}(s_j) \in \mathcal{F}_b$ ($1 \leq j \leq l$). Therefore $\text{abstract}(s_j) = \text{init}$ and it is sufficient to choose $r_j = \text{init}$ to obtain $\varphi'_i \vdash_{\varepsilon_a}^{r_j} \text{abstract}(s_j)$ ($1 \leq j \leq l$). Furthermore we already know that all s_j are i -good and therefore we can conclude that $C[s_1, \dots, s_l]$ is i -good.

We also need to show that this transition in the conclusion works, i.e. that $x\sigma'_i =_{\varepsilon_a} t\sigma'_{i-1}$. We know that $\text{collapse}(x\sigma_i) =_{\varepsilon} \text{collapse}(t\sigma_{i-1})$ (by the transition in the hypothesis). We also have that $x\sigma'_i = \text{abstract}(x\sigma_i)$ and that $t\sigma'_{i-1} = \text{abstract}(t\sigma_{i-1})$. Using Lemma 10 and Lemma 9, we immediately conclude.

- 3) if the action was a test $[s = t]$, then ξ'_i is the empty string. We choose ξ'_i also as the empty string.

As $\varphi_i = \varphi_{i-1}$ and $\sigma_i = \sigma_{i-1}$, it is sufficient to show that $s\sigma'_{i-1} =_{\varepsilon_a} t\sigma'_{i-1}$. But $s\sigma'_{i-1} = \text{abstract}(s\sigma_{i-1})$ and $t\sigma'_{i-1} = \text{abstract}(t\sigma_{i-1})$ by the definition of σ'_{i-1} and abstract .

By Lemma 10 we have that $\text{abstract}(\text{collapse}(s\sigma_{i-1})) =_{\varepsilon_a} \text{abstract}(s\sigma_{i-1})$ and analogously for t .

We conclude by Lemma 9 that $\text{collapse}(t\sigma_{i-1}) =_{\varepsilon} \text{collapse}(s\sigma_{i-1})$ (which we know because we have the corresponding transition in the hypothesis) implies $\text{abstract}(\text{collapse}(t\sigma_{i-1})) =_{\varepsilon_a} \text{abstract}(\text{collapse}(s\sigma_{i-1}))$.

- 4) if the action was an output $\text{out}(t)$, we have that $\varphi_i = \varphi_{i-1} \cup \{w_i \mapsto \text{collapse}(t\sigma_{i-1})\}$ and that ξ'_i is the empty string. We choose ξ'_i also as the empty string.

We first have to establish that $\varphi_i(w_i) = \text{collapse}(t\sigma_{i-1})$ is an i -good term, which is exactly the same as in the case of the assignment $x := t$ (see second item above).

We also have to establish that this transition works in the conclusion, i.e. that $\varphi'_i(w_i) =_{\varepsilon_a} t\sigma'_{i-1}$ knowing that $\varphi_i(w_i) = \text{collapse}(t\sigma_{i-1})$ (i.e. that the transition works in the hypothesis). We can conclude by Lemma 10 ($\varphi'_i(w_i) = \text{abstract}(\varphi_i(w_i)) = \text{abstract}(\text{collapse}(t\sigma_{i-1})) =_{\varepsilon_a} \text{abstract}(t\sigma_{i-1}) = t\sigma'_{i-1}$).

- 5) if the action was an input $\text{in}(x)$, we have that $\sigma_i = \sigma_{i-1} \cup \{x \mapsto \text{collapse}(r\varphi_{i-1})\}$ for some recipe r such that $\xi'_i = r$.

We prove by induction on r that $\text{collapse}(r\varphi_{i-1})$ is an i -good term and at the same time we construct a pure \mathcal{F}_a -recipe r' such that $r'\varphi'_{i-1} =_{\varepsilon_a} \text{abstract}(r\varphi_{i-1})$. We choose ξ'_i to be r' . This means in particular that the transition in the hypothesis will work ($x\sigma'_i =_{\varepsilon_a} r'\varphi'_{i-1}$).

Therefore all we need is the proof of i -goodness and the construction of r' by induction.

- a) base case: If r is a variable w , we have that $r\varphi_{i-1}$ is i -good by the induction hypothesis (the outer induction). We choose $r' = w$ and we obtain $r'\varphi'_{i-1} = w\varphi'_{i-1} = \text{abstract}(w\varphi_{i-1}) = \text{abstract}(r\varphi_{i-1})$.
- b) Let $r\varphi_{i-1} = C[[t_1, \dots, t_k]]$. Let c be such that $\text{root}(C) \in \mathcal{F}_c$.

Each t_j ($1 \leq j \leq k$) is such that $t_j = r_j\varphi_{i-1}$ for some recipe $r_j \subset r$ or an alien subterm of $\varphi_{i-1}(w)$ for some variable $w \in \text{dom}(\varphi_{i-1})$. In the first case we know that $t_j = \text{collapse}(t_j)$ is an i -good term by the outer induction hypothesis (t_j is an alien subterm of $\varphi_{i-1}(w)$ for some $w \in \text{dom}(\varphi_{i-1})$) and in the second case we that $\text{collapse}(t_j)$ is an i -good term by the inner induction hypothesis. Similarly, there exist pure \mathcal{F}_a -recipes over φ'_{i-1} for $\text{abstract}(\text{collapse}(t_j))$.

Let $\text{collapse}(t_j) = C_j[[t_{k_j}^j, \dots, t_{1_j}^j]]$ and let $C'_j = C_j$ if $\text{root}(C_j) \in \mathcal{F}_c$ and let $C'_j = _$ otherwise ($1 \leq j \leq k$). Let $C' = C[C_1, \dots, C_k]$. As $t_{k_j}^j$ are the alien subterms of an i -good term, it follows that $t_{k_j}^j$ are i -good terms ($1 \leq j \leq k$, $1 \leq j' \leq k_j$) and there exist pure \mathcal{F}_a -recipes over φ'_{i-1} for $\text{abstract}(t_{k_j}^j)$.

Then $r\varphi_{i-1} = C'[[s_1, \dots, s_l]]$ where each s_j ($1 \leq j \leq l$) is either some $t_{j'}$ ($1 \leq j' \leq k$) or some $t_{z'}^{j'}$ ($1 \leq j' \leq k$, $1 \leq z \leq k_{j'}$). In either case s_j ($1 \leq j \leq l$) are i -good terms and there exist pure \mathcal{F}_a -recipes over φ'_{i-1} for $\text{abstract}(s_j)$.

As $\text{collapse}(r\varphi_{i-1})$ is

- i) either some s_j , in which case we conclude directly that it is an i -good term and there is a pure \mathcal{F}_a -recipe over φ'_{i-1} for $\text{abstract}(s_j)$
- ii) or it is $C'[[s_1, \dots, s_l]]$, in which it is also easy to establish that $C'[[s_1, \dots, s_l]]$ is an i -good term (its alien subterms are i -good terms and there are pure \mathcal{F}_a -recipes over φ'_{i-1} for their abstractions).

We also need to show that there is a pure \mathcal{F}_a -recipe over φ'_{i-1} for $C'[[s_1, \dots, s_l]]$. In the case $\text{root}(C') \in \mathcal{F}_b$, we simply choose the recipe $r' = \text{init}$. Otherwise, if $\text{root}(C') = \text{root}(r) = f \in \mathcal{F}_a$ and $r = f(r_1, \dots, r_m)$, we let $r' = f(r'_1, \dots, r'_m)$. It is then easy to show by induction that $r'\varphi'_{i-1} =_{E_a} \text{abstract}(r\varphi_{i-1})$. ■

C. Tagging

Lemma 8: If P is a protocol over $\mathcal{F}_{\text{enc,h}}$, then $P \models_{E_{\text{enc}}} \text{Secret}(x)$ implies $[[P^c]] \models_{E_{\text{enc}} \cup E_c} \text{Secret}(x)$.

Proof:

We consider w.l.o.g. that $c = a$. We show that if $[[P^a]] \not\models_{E_{\text{enc}} \cup E_a} \text{Secret}(x)$ then $P \not\models_{E_{\text{enc}}} \text{Secret}(x)$. Then

there exists a trace $([[P^a]], \emptyset, \emptyset) \xrightarrow{r_1 \dots r_n}_{E_{\text{enc}} \cup E_a} (Q, \varphi, \sigma)$ such that $\varphi \vdash_{E_{\text{enc}} \cup E_a} x\sigma$.

We construct a trace

$$(P, \emptyset, \emptyset) \xrightarrow{r'_1 \dots r'_n}_{E_{\text{enc}}} (Q', \varphi', \sigma') \quad (19)$$

such that $\varphi' \vdash_{E_{\text{enc}}} x\sigma'$.

For any term t (including r_1, \dots, r_n), we let t' be the term obtained from t by removing any reference of tag and untag . It is formally defined inductively as follows:

- 1) $\text{enc}(t_1, t_2)' = \text{enc}(t'_1, t'_2)$
- 2) $\text{dec}(t_1, t_2)' = \text{dec}(t'_1, t'_2)$
- 3) $h(t_1)' = h(t'_1)$
- 4) $\text{init}' = \text{init}$
- 5) $\text{tag}(t_1)' = t'_1$
- 6) $\text{untag}(t_1)' = t'_1$
- 7) $t' = t$ if t is a variable or a name

We let $\sigma'(x) = \sigma(x)'$ and $\varphi'(w) = \varphi(w)'$.

To prove Equation 19, it is sufficient to establish that:

- 1) (tests work) $\mathcal{H}(s^a)\sigma =_{E_{\text{enc}} \cup E_a} \mathcal{H}(t^a)\sigma$ implies that $s\sigma' =_{E_{\text{enc}}} t\sigma'$
- 2) (inputs work) $x\sigma =_{E_{\text{enc}} \cup E_c} r\varphi$ implies that $x\sigma' =_{E_{\text{enc}}} r'\varphi'$
- 3) (outputs work) $w_i\varphi =_{E_{\text{enc}} \cup E_c} \mathcal{H}(t^a)\sigma$ implies that $w_i\varphi' =_{E_{\text{enc}}} t\sigma'$

Let R be the convergent term rewriting system obtained by orienting E_{enc} from left to right and let R_c be the convergent term rewriting system obtained by orienting the equations of $E_{\text{enc}} \cup E_a$ from left to right. We have that $s =_E t$ (resp. $s =_{E_{\text{enc}} \cup E_a} t$) iff $s \downarrow_R = t \downarrow_R$ (resp. $s \downarrow_{R_c} = t \downarrow_{R_c}$), where $t \downarrow_R$ represents the normal form of t with respect to R .

As $(s^c)' = s$, $(t^c)' = t$, $x' = x$ and $w'_i = w_i$, all of the above are instances of the more general implication $s =_{E_{\text{enc}} \cup E_c} t$ implies that $s' =_{E_{\text{enc}}} t'$. This implication is easy to prove, since $(s \downarrow_{R_c})' = s' \downarrow_R$ (proof by induction on s). ■

Lemma 7: Let P and Q be linear processes over $\mathcal{F}_{\text{enc,h}}$. Let W be an arbitrary interleaving of P^a and Q^b and let $R = [[W]]$. If R reveals x then W reveals x .

Proof:

We transform any attack trace on R into an attack trace on W . Let

$$(R, \emptyset, \emptyset) \xrightarrow{r_1 \dots r_k}_{E_{\text{enc}}} (R', \varphi, \sigma) \quad (20)$$

be such that $\varphi \vdash \sigma(x)$.

We show that

$$(W, \emptyset, \emptyset) \xrightarrow{r'_1 \dots r'_k}_{E_{\text{enc}}} (W', \varphi', \sigma') \quad (21)$$

such that $\varphi' \vdash \sigma'(x)$.

For $c \in \{a, b, d\}$, let $\mathcal{F}'_c = \{\text{enc}_c, \text{dec}_c, h_c\}$ and $E'_c = \{\text{dec}_c(\text{enc}_c(x, y), y) = x\}$. For $c' \in \{a', b'\}$, let $\mathcal{F}_{c'} = \{\text{tag}_{c'}, \text{untag}_{c'}\}$ and $E_{c'} = \{\text{untag}_{c'}(\text{tag}_{c'}(x)) = x\}$.

We show that Equation 21 holds in the signature $\mathcal{F}^0 = \mathcal{F}'_a \cup \mathcal{F}'_b \cup \mathcal{F}_{a'} \cup \mathcal{F}_{b'} \cup \mathcal{F}'_d$ and in the equational theory $E^0 = E'_a \cup E'_b \cup E_{a'} \cup E_{b'} \cup E'_d$. We then conclude by Lemma 6.

We consider the following transformation $\lceil _ \rceil$ on terms (in the following, c ranges over $\{a, b\}$ (but not d) and c' ranges over $\{a', b'\}$ (but not d' , which does not exist)):

$$\begin{aligned}
\lceil \text{tag}_c(t) \rceil &= \text{tag}_{c'}(\lceil t \rceil) \\
\lceil \text{enc}(t_1, t_2) \rceil &= \text{enc}_c(t_3, \lceil t_2 \rceil) \text{ if } \lceil t_1 \rceil =_{\mathbb{E}} \text{tag}_{c'}(t_3) \\
&= \text{enc}_d(\lceil t_1 \rceil, \lceil t_2 \rceil) \text{ otherwise} \\
\lceil \text{dec}(t_1, t_2) \rceil &= \text{tag}_{c'}(\text{dec}_c(\lceil t_1 \rceil, \lceil t_2 \rceil)) \text{ if } \lceil t_1 \rceil =_{\mathbb{E}} \text{enc}_c(_, \lceil t_2 \rceil) \\
&= \text{dec}_d(\lceil t_1 \rceil, \lceil t_2 \rceil) \text{ otherwise} \\
\lceil \text{untag}_c(t) \rceil &= \text{untag}_{c'}(\lceil t \rceil) \\
\lceil h(t) \rceil &= h_c(t_0) \text{ if } \lceil t \rceil =_{\mathbb{E}} \text{tag}_{c'}(t_0) \\
&= h_d(\lceil t \rceil) \text{ otherwise} \\
\lceil u \rceil &= u \text{ for a name or variable } u
\end{aligned}$$

Note that if $s =_{\mathbb{E}} t$ then

$$\lceil s \rceil =_{\mathbb{E}^0} \lceil t \rceil. \quad (22)$$

We let $\sigma' = \lceil \sigma \rceil$ and $\varphi' = \lceil \varphi \rceil$. We next construct by induction on the number of transitions in Equation 21 the recipes r'_1, \dots, r'_k such that: for all subterms of σ' and φ' of the form $\text{tag}_{c'}(t)$ there exists a recipe r for t :

- 1) if the current action is an input action $\mathbf{in}(x)$, we transform the recipe r used for the same transition in Equation 20 into a recipe $\lceil r \rceil$ as follows:

$$\begin{aligned}
\lceil \text{tag}_c(r) \rceil &= \text{tag}_{c'}(\lceil r \rceil) \\
\lceil \text{enc}(r_1, r_2) \rceil &= \text{enc}_c(r_3, \lceil r_2 \rceil) \text{ if } \lceil r_1 \rceil \varphi' =_{\mathbb{E}} \text{tag}_{c'}(t_3) \\
&\quad \text{and where } r_3 \text{ is a recipe for } t_3 \\
&= \text{enc}_d(\lceil r_1 \rceil, \lceil r_2 \rceil) \text{ otherwise} \\
\lceil \text{dec}(r_1, r_2) \rceil &= \text{tag}_{c'}(\text{dec}_c(\lceil r_1 \rceil, \lceil r_2 \rceil)) \\
&\quad \text{if } \lceil r_1 \rceil \varphi' =_{\mathbb{E}} \text{enc}_c(_, \lceil r_2 \rceil \varphi') \\
&= \text{dec}_d(\lceil r_1 \rceil, \lceil r_2 \rceil) \text{ otherwise} \\
\lceil \text{untag}_c(t) \rceil &= \text{untag}_{c'}(\lceil r \rceil) \\
\lceil h(t) \rceil &= h_c(r_0) \text{ if } \lceil t \rceil =_{\mathbb{E}} \text{tag}_{c'}(t_0) \\
&\quad \text{and where } r_0 \text{ is a recipe for } t_0 \\
&= h_d(\lceil t \rceil) \text{ otherwise} \\
\lceil w \rceil &= w \text{ for a variable } w
\end{aligned}$$

We know that $r\varphi =_{\mathbb{E}} x\sigma$ and we have to show that $\lceil r \rceil \varphi' =_{\mathbb{E}^0} x\sigma'$. By Equation 22 we have that $\lceil r\varphi \rceil =_{\mathbb{E}^0} \lceil x\sigma \rceil$. But $\lceil x\sigma \rceil = x\sigma'$ by definition. Therefore, to establish our conclusion, it is sufficient to show that $\lceil r\varphi \rceil =_{\mathbb{E}^0} \lceil r \rceil \varphi'$. But this follows immediately by induction on r .

- 2) if the current action is an output action $\mathbf{out}(t)$ (assume that t is from \mathcal{F}_c) we know that $\text{tests}^c(t)$ passed in Equation 20 and that $w\varphi =_{\mathbb{E}} \mathcal{H}(t)\sigma$. We have to show that $w\varphi' =_{\mathbb{E}^0} t\sigma'$. By Equation 22, we have that $w\varphi' = \lceil w\varphi \rceil =_{\mathbb{E}^0} \lceil \mathcal{H}(t)\sigma \rceil$. To establish the conclusion, it is sufficient therefore to show that $t\sigma' = \lceil \mathcal{H}(t)\sigma \rceil$. This follows by induction on t (for the case of untagged decryption, we use that $\text{tests}^c(t)$ work).
- 3) if the current action is a test $[s = t]$ (assume that $[s = t]$ is from \mathcal{F}_c), we know that $\mathcal{H}(s)\sigma =_{\mathbb{E}} \mathcal{H}(t)\sigma$ and that $\text{tests}^c(s)$ and $\text{tests}^c(t)$ work. We have to show that $s\sigma' =_{\mathbb{E}^0} t\sigma'$. It is sufficient to show that $\lceil H(u)\sigma \rceil =_{\mathbb{E}^0} u\sigma'$ when $\text{tests}^c(u)$ work (by induction on u), since then we can use the equality for $u \in \{s, t\}$ and we can conclude by Equation 22.

- 4) if the current action is an assignment $x := t$ (assume it comes from \mathcal{F}_c) we have that $\text{tests}^c(t)$ work and that $x\sigma =_{\mathbb{E}} \mathcal{H}(t)\sigma$. We have to show that $x\sigma' =_{\mathbb{E}^0} t\sigma'$. We have from the previous item that $\lceil \mathcal{H}(t)\sigma \rceil =_{\mathbb{E}^0} t\sigma'$, and we conclude by Equation 22.
- 5) if the current action is a new νx , we have to show that $x\sigma' = \lceil x\sigma \rceil = x\sigma$ is a fresh name, which is immediate. ■