Composing security protocols: from confidentiality to privacy *

Myrto Arapinis[1], Vincent Cheval[2,3], and Stéphanie Delaune[4]

[1] School of Informatics, University of Edinburgh, UK
[2] LORIA, CNRS, France
[3] School of Computing, University of Kent, UK
[4] LSV, CNRS & ENS Cachan, France

Abstract. Security protocols are used in many of our daily-life applications, and our privacy largely depends on their design. Formal verification techniques have proved their usefulness to analyse these protocols, but they become so complex that modular techniques have to be developed. We propose several results to safely compose security protocols. We consider arbitrary primitives modeled using an equational theory, and a rich process algebra close to the applied pi calculus.
Relying on these composition results, we derive some security properties on a protocol from the security analysis performed on each of its sub-protocols individually. We consider parallel composition and the case of key-exchange protocols. Our results apply to deal with confidentiality but also privacy-type properties (*e.g.* anonymity) expressed using a notion of equivalence. We illustrate the usefulness of our composition results on protocols from the 3G phone application and electronic passport.

1 Introduction

Privacy means that one can control when, where, and how information about oneself is used and by whom, and it is actually an important issue in many modern applications. For instance, nowadays, it is possible to wave an electronic ticket, a building access card, a government-issued ID, or even a smartphone in front of a reader to go through a gate, or to pay for some purchase. Unfortunately, as often reported by the media, this technology also makes it possible for anyone to capture some of our personal information. To secure the applications mentioned above and to protect our privacy, some specific cryptographic protocols are deployed. For instance, the 3G telecommunication application allows one to send SMS encrypted with a key that is established with the *AKA* protocol [2]. The aim of this design is to provide some security guarantees: *e.g.* the SMS exchanged between phones should remain confidential from third parties.

Since security protocols are notoriously difficult to design and analyse, formal verification techniques are important. These techniques have become mature and

* The research leading to these results has received funding from the project ProSecure (ERC grant agreement n° 258865), and the ANR project VIP n° 11 JS02 006 01.

have achieved success. For instance, a flaw has been discovered in the Single-Sign-On protocol used by Google Apps [6], and several verification tools are nowadays available (*e.g.* ProVerif [9], the AVANTSSAR platform [7]). These tools perform well in practice, at least for standard security properties (*e.g.* secrecy, authentication). Regarding privacy properties, the techniques and tools are more recent. Most of the verification techniques are only able to analyse a bounded number of sessions and consider a quite restrictive class of protocols (*e.g.* [18]). A different approach consists in analysing a stronger notion of equivalence, namely *diff-equivalence*. In particular, ProVerif implements a semi-decision procedure for checking diff-equivalence [9].

Security protocols used in practice are more and more complex and it is difficult to analyse them altogether. For example, the UMTS standard [2] specifies tens of sub-protocols running concurrently in 3G phone systems. While one may hope to verify each protocol in isolation, it is however unrealistic to expect that the whole application will be checked relying on a unique automatic tool. Existing tools have their own specificities that prevent them to be used in some cases. Furthermore, most of the techniques do not scale up well on large systems, and sometimes the ultimate solution is to rely on a manual proof. It is therefore important that the protocol under study is as small as possible.

Related work. There are many results studying the composition of security protocols in the symbolic model [15, 13, 12], as well as in the computational model [8, 16] in which the so-called UC (universal composability) framework has been first developed before being adapted in the symbolic setting [10]. Our result belongs to the first approach. Most of the existing composition results are concerned with trace-based security properties, and in most cases only with secrecy (stated as a reachability property), *e.g.* [15, 13, 12, 14]. They are quite restricted in terms of the class of protocols that can be composed, *e.g.* a fixed set of cryptographic primitives and/or no else branch. Lastly, they often only consider parallel composition. Some notable exceptions are the results presented in [17, 14, 12]. This paper is clearly inspired from the approach developed in [12].

Regarding privacy-type properties, very few composition results exist. In a previous work [4], we considered parallel composition only. More precisely, we identified sufficient conditions under which protocols can "safely" be executed in parallel as long as they have been proved secure in isolation. This composition theorem was quite general from the point of view of the cryptographic primitives allowed. We considered arbitrary primitives that can be modelled by a set of equations, and protocols may share some standard primitives provided they are tagged differently. We choose to reuse this quite general setting in this work, but our goal is now to go beyond parallel composition. We want to extend the composition theorem stated in [4] to allow a modular analysis of protocols that use other protocols as sub-programs as it happens in key-exchange protocols.

Our contributions. Our main goal is to analyse privacy-type properties in a modular way. These security properties are usually expressed as equivalences between processes. Roughly, two processes P and Q are equivalent ($P \approx Q$) if, however they behave, the messages observed by the attacker are indistinguishable. Actually, it is well-known that:

$$\text{if } P_1 \approx P_2 \text{ and } Q_1 \approx Q_2 \text{ then } P_1 \mid P_2 \approx Q_1 \mid Q_2.$$

However, this parallel composition result works because the processes that are composed are disjoint (*e.g.* they share no key). In this paper, we want to go beyond parallel composition which was already considered in [4]. In particular, we want to capture the case where a protocol uses a sub-protocol to establish some keys. To achieve this, we propose several theorems that state the conditions that need to be satisfied so that the security of the whole protocol can be derived from the security analysis performed on each sub-protocol in isolation. They are all derived from a generic composition result that allows one to map a trace of the composed protocol into a trace of the disjoint case (protocol where the sub-protocols do not share any data), and conversely. This generic result can be seen as an extension of the result presented in [12] where only a mapping from the shared case to the disjoint case is provided (but not the converse).

We also extend [12] by considering a richer process algebra. In particular, we are able to deal with protocols with else branches and to compose protocols that both rely on asymmetric primitives (*i.e.* asymmetric encryption and signature).

Outline. We present our calculus in Section 2. It can be seen as an extension of the applied pi calculus with an assignment construction. This will allow us to easily express the sharing of some data (*e.g.* session keys) between sub-protocols. In Section 3, we present a first composition result to deal with confidentiality properties. The purpose of this section is to review the difficulties that arise when composing security protocols even in a simple setting. In Section 4, we go beyond parallel composition, and we consider the case of key-exchange protocols. We present in Section 5 some additional difficulties that arise when we want to consider privacy-type properties expressed using trace equivalence. In Section 6, we present our composition results for privacy-type properties. We consider parallel composition as well as the case of key-exchange protocols. In Section 7, we illustrate the usefulness of our composition results on protocols from the 3G phone application, as well as on protocols from the e-passport application. We show how to derive some security guarantees from the analysis performed on each sub-protocol in isolation. The full version of this paper as well as the ProVerif models of our case studies can be found at `http://www.loria.fr/~chevalvi/other/compo/`.

2 Models for security protocols

Our calculus is close to the applied pi calculus [3]. We consider an assignment operation to make explicit the data that are shared among different processes.

2.1 Messages

As usual in this kind of models, messages are modelled using an abstract term algebra. We assume an infinite set of *names* $\mathcal{N}$ of *base type* (used for representing keys, nonces, ...) and a set $\mathcal{C}h$ of names of *channel type*. We also consider a set of *variables* $\mathcal{X}$, and a signature Σ consisting of a finite set of *function symbols*. We rely on a sort system for terms. The details of the sort system are unimportant, as long as the base type differs from the channel type, and we suppose that function symbols only operate on and return terms of base type.

Terms are defined as names, variables, and function symbols applied to other terms. The set of terms built from $\mathsf{N} \subseteq \mathcal{N} \cup \mathcal{C}h$, and $\mathsf{X} \subseteq \mathcal{X}$ by applying function symbols in Σ (respecting sorts and arities) is denoted by $\mathcal{T}(\Sigma, \mathsf{N} \cup \mathsf{X})$. We write $fv(u)$ (resp. $fn(u)$) for the set of variables (resp. names) occurring in a term u. A term u is *ground* if it does not contain any variable, *i.e.* $fv(u) = \emptyset$.

The algebraic properties of cryptographic primitives are specified by the means of an *equational theory* which is defined by a finite set E of equations $u = v$ with $u, v \in \mathcal{T}(\Sigma, \mathcal{X})$, *i.e.* u, v do not contain names. We denote by $=_{\mathsf{E}}$ the smallest equivalence relation on terms, that contains E and that is closed under application of function symbols and substitutions of terms for variables.

Example 1. Consider the signature $\Sigma_{\mathsf{DH}} = \{\mathsf{aenc}, \mathsf{adec}, \mathsf{pk}, \mathsf{g}, \mathsf{f}, \langle\ \rangle, \mathsf{proj}_1, \mathsf{proj}_2\}$. The function symbols adec, aenc of arity 2 represent asymmetric decryption and encryption. We denote by $\mathsf{pk}(sk)$ the public key associated to the private key sk. The two function symbols f of arity 2, and g of arity 1 are used to model the Diffie-Hellman primitives, whereas the three remaining symbols are used to model pairs. The equational theory E_{DH} is defined by:

$$\mathsf{E}_{\mathsf{DH}} = \left\{ \begin{array}{ll} \mathsf{proj}_1(\langle x, y\rangle) = x & \mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x \\ \mathsf{proj}_2(\langle x, y\rangle) = y & \mathsf{f}(\mathsf{g}(x), y) = \mathsf{f}(\mathsf{g}(y), x) \end{array} \right.$$

Let $u_0 = \mathsf{aenc}(\langle n_A, \mathsf{g}(r_A)\rangle, \mathsf{pk}(sk_B))$. We have that:

$$\mathsf{f}(\mathsf{proj}_2(\mathsf{adec}(u_0, sk_B)), r_B) =_{\mathsf{E}_{\mathsf{DH}}} \mathsf{f}(\mathsf{g}(r_A), r_B) =_{\mathsf{E}_{\mathsf{DH}}} \mathsf{f}(\mathsf{g}(r_B), r_A).$$

2.2 Processes

As in the applied pi calculus, we consider *plain processes* as well as *extended processes* that represent processes having already evolved by *e.g.* disclosing some terms to the environment. *Plain processes* are defined by the following grammar:

$$\begin{array}{llll} P, Q := 0 & \text{null} & P \mid Q & \text{parallel} \\ \quad \mathtt{new}\ n.P & \text{restriction} & !P & \text{replication} \\ \quad [x := v].P & \text{assignment} & \mathtt{if}\ \varphi\ \mathtt{then}\ P\ \mathtt{else}\ Q & \text{conditional} \\ \quad \mathtt{in}(c, x).P & \text{input} & \mathtt{out}(c, v).Q & \text{output} \end{array}$$

where c is a name of channel type, φ is a conjunction of tests of the form $u_1 = u_2$ where u_1, u_2 are terms of base type, x is a variable of base type, v is a term of base type, and n is a name of any type. We consider an assignment operation that instantiates x with a term v. Note that we consider private channels but we do not allow channel passing. For the sake of clarity, we often omit the null process, and when there is no "else", it means "else 0".

Names and variables have scopes, which are delimited by restrictions, inputs, and assignment operations. We write $fv(P)$, $bv(P)$, $fn(P)$ and $bn(P)$ for the sets of *free* and *bound variables*, and *free* and *bound names* of a plain process P.

Example 2. Let $P_{\mathsf{DH}} = \mathtt{new}\, sk_A.\mathtt{new}\, sk_B.(P_A \mid P_B)$ a process that models a Diffie-Hellman key exchange protocol:

– $P_A \stackrel{\mathsf{def}}{=} \mathtt{new}\, r_A.\mathtt{new}\, n_A.\mathtt{out}(c, \mathsf{aenc}(\langle n_A, \mathsf{g}(r_A)\rangle, \mathsf{pk}(sk_B))).\mathtt{in}(c, y_A).$
$\quad \mathtt{if}\ \ \mathsf{proj}_1(\mathsf{adec}(y_A, sk_A)) = n_A\ \ \mathtt{then}\ [x_A := \mathsf{f}(\mathsf{proj}_2(\mathsf{adec}(y_A, sk_A)), r_A)].0$
– $P_B \stackrel{\mathsf{def}}{=} \mathtt{new}\, r_B.\mathtt{in}(c, y_B).\mathtt{out}(c, \mathsf{aenc}(\langle \mathsf{proj}_1(\mathsf{adec}(y_B, sk_B)), \mathsf{g}(r_B)\rangle, \mathsf{pk}(sk_A))).$
$\quad [x_B := \mathsf{f}(\mathsf{proj}_2(\mathsf{adec}(y_B, sk_B)), r_B)].0$

The process P_A generates two fresh random numbers r_A and n_A, sends a message on the channel c, and waits for a message containing the nonce n_A in order to compute his own view of the key that will be stored in x_A. The process P_B proceeds in a similar way and stores the computed value in x_B.

Extended processes add a set of restricted names $\mathcal{E}$ (the names that are *a priori* unknown to the attacker), a sequence of messages Φ (corresponding to the messages that have been sent so far on public channels) and a substitution σ which is used to store the messages that have been received as well as those that have been stored in assignment variables.

Definition 1. *An* extended process *is a tuple* $(\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ *where* $\mathcal{E}$ *is a set of names that represents the names that are restricted in* $\mathcal{P}$*,* Φ *and* σ*;* $\mathcal{P}$ *is a multiset of* plain processes *where null processes are removed and such that* $fv(\mathcal{P}) \subseteq \mathrm{dom}(\sigma)$*;* $\Phi = \{w_1 \rhd u_1, \dots, w_n \rhd u_n\}$ *and* $\sigma = \{x_1 \mapsto v_1, \dots, x_m \mapsto v_m\}$ *are substitutions where* $u_1, \dots, u_n, v_1, \dots, v_m$ *are ground terms, and* $w_1, \dots, w_n$*,* $x_1, \dots, x_m$ *are variables.*

For the sake of simplicity, we assume that extended processes are *name and variable distinct*, *i.e.* a name (resp. variable) is either free or bound, and in the latter case, it is at most bound once. We write $(\mathcal{E}; P; \Phi)$ instead of $(\mathcal{E}; P; \Phi; \emptyset)$.

The semantics is given by a set of labelled rules that allows one to reason about processes that interact with their environment (see Figure 1). This defines the relation $\xrightarrow{\ell}$ where ℓ is either an input, an output, or a silent action τ. The relation $\xrightarrow{\mathsf{tr}}$ where tr denotes a sequence of labels is defined in the usual way whereas the relation $\xRightarrow{\mathsf{tr}'}$ on processes is defined by: $A \xRightarrow{\mathsf{tr}'} B$ if, and only if, there exists a sequence tr such that $A \xrightarrow{\mathsf{tr}} B$ and tr' is obtained by erasing all occurrences of the silent action τ in tr.

Example 3. Let $\Phi_{\mathsf{DH}} \stackrel{\mathsf{def}}{=} \{w_1 \rhd \mathsf{pk}(sk_A), w_2 \rhd \mathsf{pk}(sk_B)\}$. We have that:

$$(\{sk_A, sk_B\}; P_A \mid P_B; \Phi_{\mathsf{DH}}) \xRightarrow{\nu w_3.\mathtt{out}(c,w_3).\mathtt{in}(c,w_3).\nu w_4.\mathtt{out}(c,w_4).\mathtt{in}(c,w_4)} (\mathcal{E}; \emptyset; \Phi_{\mathsf{DH}} \uplus \Phi; \sigma \cup \sigma')$$

where $\Phi =_{\mathsf{E_{DH}}} \{w_3 \rhd u_0, w_4 \rhd \mathsf{aenc}(\langle n_A, \mathsf{g}(r_B)\rangle, pk_A)\}$, $\mathcal{E} = \{sk_A, sk_B, r_A, r_B, n_A\}$, $\sigma =_{\mathsf{E_{DH}}} \{y_A \mapsto \mathsf{aenc}(\langle n_A, \mathsf{g}(r_B)\rangle, pk_A),\ y_B \mapsto \mathsf{aenc}(\langle n_A, \mathsf{g}(r_A)\rangle, pk_B)\}$, and lastly $\sigma' =_{\mathsf{E_{DH}}} \{x_A \mapsto \mathsf{f}(\mathsf{g}(r_B), r_A),\ x_B \mapsto \mathsf{f}(\mathsf{g}(r_A), r_B)\}$. We used pk_A (resp. pk_B) as a shorthand for $\mathsf{pk}(sk_A)$ (resp. $\mathsf{pk}(sk_B)$).

$$(\mathcal{E}; \{\texttt{if } \varphi \texttt{ then } Q_1 \texttt{ else } Q_2\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau} (\mathcal{E}; Q_1 \uplus \mathcal{P}; \Phi; \sigma) \quad \text{(THEN)}$$
$$\text{if } u\sigma =_{\mathsf{E}} v\sigma \text{ for each } u = v \in \varphi$$
$$(\mathcal{E}; \{\texttt{if } \varphi \texttt{ then } Q_1 \texttt{ else } Q_2\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau} (\mathcal{E}; Q_2 \uplus \mathcal{P}; \Phi; \sigma) \quad \text{(ELSE)}$$
$$\text{if } u\sigma \neq_{\mathsf{E}} v\sigma \text{ for some } u = v \in \varphi$$
$$(\mathcal{E}; \{\texttt{out}(c,u).Q_1; \texttt{in}(c,x).Q_2\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau} (\mathcal{E}; Q_1 \uplus Q_2 \uplus \mathcal{P}; \Phi; \sigma \cup \{x \mapsto u\sigma\}) \quad \text{(COMM)}$$
$$(\mathcal{E}; \{[x := v].Q\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau} (\mathcal{E}; Q \uplus \mathcal{P}; \Phi; \sigma \cup \{x \mapsto v\sigma\}) \quad \text{(ASSGN)}$$

$$(\mathcal{E}; \{\texttt{in}(c,z).Q\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\texttt{in}(c,M)} (\mathcal{E}; Q \uplus \mathcal{P}; \Phi; \sigma \cup \{z \mapsto u\}) \quad \text{(IN)}$$
$$\text{if } c \notin \mathcal{E},\ M\Phi = u,\ fv(M) \subseteq \mathrm{dom}(\Phi) \text{ and } fn(M) \cap \mathcal{E} = \emptyset$$
$$(\mathcal{E}; \{\texttt{out}(c,u).Q\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\nu w_i.\texttt{out}(c,w_i)} (\mathcal{E}; Q \uplus \mathcal{P}; \Phi \cup \{w_i \rhd u\sigma\}; \sigma) \quad \text{(OUT-T)}$$
$$\text{if } c \notin \mathcal{E},\ u \text{ is a term of base type, and } w_i \text{ is a variable such that } i = |\Phi| + 1$$

$$(\mathcal{E}; \{\texttt{new } n.Q\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau} (\mathcal{E} \cup \{n\}; Q \uplus \mathcal{P}; \Phi; \sigma) \quad \text{(NEW)}$$
$$(\mathcal{E}; \{!Q\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau} (\mathcal{E}; \{!Q; Q\rho\} \uplus \mathcal{P}; \Phi; \sigma) \quad \text{(REPL)}$$
$$\rho \text{ is used to rename } bv(Q)/bn(Q) \text{ with fresh variables/names}$$
$$(\mathcal{E}; \{P_1 \mid P_2\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau} (\mathcal{E}; \{P_1, P_2\} \uplus \mathcal{P}; \Phi; \sigma) \quad \text{(PAR)}$$

where n is a name, c is a name of channel type, u, v are terms of base type, and x, z are variables of base type.

Fig. 1. Semantics of extended processes

2.3 Process equivalences

We are particularly interested in security properties expressed using a notion of equivalence such as those studied in *e.g.* [5, 11]. For instance, the notion of *strong unlinkability* can be formalized using an equivalence between two situations: one where each user can execute the protocol multiple times, and one where each user can execute the protocol at most once.

We consider here the notion of *trace equivalence*. Intuitively, two protocols P and Q are in trace equivalence, denoted $P \approx Q$, if whatever the messages they received (built upon previously sent messages), the resulting sequences of messages sent on public channels are indistinguishable from the point of view of an outsider. Given an extended process A, we define its set of traces as follows:

$$\mathsf{trace}(A) = \{(\mathsf{tr}, \texttt{new } \mathcal{E}.\Phi) \mid A \xRightarrow{\mathsf{tr}} (\mathcal{E}; \mathcal{P}; \Phi; \sigma) \text{ for some process } (\mathcal{E}; \mathcal{P}; \Phi; \sigma)\}.$$

The sequence of messages Φ together with the set of restricted names $\mathcal{E}$ (those unknown to the attacker) is called the *frame*.

Definition 2. *We say that a term u is* deducible *(modulo E) from a frame $\phi = \texttt{new}\mathcal{E}.\Phi$, denoted $\texttt{new}\mathcal{E}.\Phi \vdash u$, when there exists a term M (called a* recipe*) such that $fn(M) \cap \mathcal{E} = \emptyset$, $fv(M) \subseteq \mathrm{dom}(\Phi)$, and $M\Phi =_{\mathsf{E}} u$.*

Two frames are indistinguishable when the attacker cannot detect the difference between the two situations they represent.

Definition 3. *Two frames ϕ_1 and ϕ_2 with $\phi_i = \mathtt{new}\mathcal{E}.\Phi_i$ ($i \in \{1,2\}$) are* statically equivalent*, denoted by $\phi_1 \sim \phi_2$, when* $\mathrm{dom}(\Phi_1) = \mathrm{dom}(\Phi_2)$*, and for all terms M, N with $fn(\{M,N\}) \cap \mathcal{E} = \emptyset$ and $fv(\{M,N\}) \subseteq \mathrm{dom}(\Phi_1)$, we have that:*

$$M\Phi_1 =_{\mathsf{E}} N\Phi_1, \text{ if and only if, } M\Phi_2 =_{\mathsf{E}} N\Phi_2.$$

Example 4. Consider $\Phi_1 = \{w_1 \rhd \mathsf{g}(r_A), w_2 \rhd \mathsf{g}(r_B), w_3 \rhd \mathsf{f}(\mathsf{g}(r_A), r_B)\}$, and $\Phi_2 = \{w_1 \rhd \mathsf{g}(r_A), w_2 \rhd \mathsf{g}(r_B), w_3 \rhd k\}$. Let $\mathcal{E} = \{r_A, r_B, k\}$. We have that $\mathtt{new}\,\mathcal{E}.\Phi_1 \sim \mathtt{new}\,\mathcal{E}.\Phi_2$ (considering the equational theory E_{DH}). This equivalence shows that the term $\mathsf{f}(\mathsf{g}(r_A), r_B)$ (the Diffie-Hellman key) is indistinguishable from a random key. This indistinguishability property holds even if the messages $\mathsf{g}(r_A)$ and $\mathsf{g}(r_B)$ have been observed by the attacker.

Two processes are trace equivalent if, whatever the messages they sent and received, their frames are in static equivalence.

Definition 4. *Let A and B be two extended processes, $A \sqsubseteq B$ if for every $(\mathsf{tr}, \phi) \in \mathit{trace}(A)$, there exists $(\mathsf{tr}', \phi') \in \mathit{trace}(B)$ such that $\mathsf{tr} = \mathsf{tr}'$ and $\phi \sim \phi'$. We say that A and B are* trace equivalent*, denoted by $A \approx B$, if $A \sqsubseteq B$ and $B \sqsubseteq A$.*

This notion of equivalence allows us to express many interesting privacy-type properties *e.g.* vote-privacy, strong versions of anonymity and/or unlinkability.

3 Composition result: a simple setting

It is well-known that even if two protocols are secure in isolation, it is not possible to compose them in arbitrary ways still preserving their security. This has already been observed for different kinds of compositions (*e.g.* parallel [15], sequential [12]) and when studying standard security properties [13] and even privacy-type properties [4]. In this section, we introduce some well-known hypotheses that are needed to safely compose security protocols.

3.1 Sharing primitives

A protocol can be used as an oracle by another protocol to decrypt a message, and then compromise the security of the whole application. To avoid this kind of interactions, most of the composition results assume that protocols do not share any primitive or allow a list of standard primitives (*e.g.* signature, encryption) to be shared as long as they are tagged in different ways. In this paper, we adopt the latter hypothesis and consider the fixed common signature:

$$\Sigma_0 = \{\mathsf{sdec}, \mathsf{senc}, \mathsf{adec}, \mathsf{aenc}, \mathsf{pk}, \langle\,,\rangle, \mathsf{proj}_1, \mathsf{proj}_2, \mathsf{sign}, \mathsf{check}, \mathsf{vk}, \mathsf{h}\}$$

equipped with the equational theory E_0, defined by the following equations:

$$\mathsf{sdec}(\mathsf{senc}(x,y),y) = x \qquad \mathsf{check}(\mathsf{sign}(x,y),\mathsf{vk}(y)) = x$$
$$\mathsf{adec}(\mathsf{aenc}(x,\mathsf{pk}(y)),y) = x \qquad \mathsf{proj}_i(\langle x_1, x_2\rangle) = x_i \text{ with } i \in \{1,2\}$$

This allows us to model symmetric/asymmetric encryption, concatenation, signatures, and hash functions. We consider a type *seed* which is a subsort of the base type that only contains names. We denote by $\mathsf{pk}(sk)$ (resp. $\mathsf{vk}(sk)$) the public key (resp. the verification key) associated to the private key sk which has to be a name of type *seed*. We allow protocols to both rely on Σ_0 provided that each application of aenc, senc, sign, and h is tagged (using disjoint sets of tags for the two protocols), and adequate tests are performed when receiving a message to ensure that the tags are correct. Actually, we consider the same tagging mechanism as the one we have introduced in [4] (see Definitions 4 and 5 in [4]). In particular, we rely on the same notation: we use the two function symbols $\mathsf{tag}/\mathsf{untag}$, and the equation $\mathsf{untag}(\mathsf{tag}(x)) = x$ to model the interactions between them. However, since we would like to be able to iterate our composition results (in order to compose *e.g.* three protocols), we consider a family of such function symbols: $\mathsf{tag}_i/\mathsf{untag}_i$ with $i \in \mathbb{N}$. Moreover, a process may be tagged using a subset of such symbols (and not only one). This gives us enough flexibility to allow different kinds of compositions, and to iterate our composition results.

Example 5. In order to compose the protocol introduced in Example 2 with another one that also relies on the primitive aenc, we may want to consider a tagged version of this protocol. The tagged version (using $\mathsf{tag}_1/\mathsf{untag}_1$) of P_B is given below (with $u = \mathsf{untag}_1(\mathsf{adec}(y_B, sk_B))$):

$$\begin{cases} \mathtt{new}\, r_B.\mathtt{in}(c, y_B). \\ \mathtt{if}\ \mathsf{tag}_1(\mathsf{untag}_1(\mathsf{adec}(y_B, sk_B))) = \mathsf{adec}(y_B, sk_B)\ \mathtt{then} \\ \mathtt{if}\ u = \langle \mathsf{proj}_1(u), \mathsf{proj}_2(u)\rangle\ \mathtt{then} \\ \mathtt{out}(c, \mathsf{aenc}(\mathsf{tag}_1(\langle \mathsf{proj}_1(u), \mathsf{g}(r_B)\rangle), \mathsf{pk}(sk_A))).[x_B := \mathsf{f}(\mathsf{proj}_2(u), r_B)].0 \end{cases}$$

The first test allows one to check that y_B is an encryption tagged with tag_1 and the second one is used to ensure that the content of this encryption is a pair as expected. Then, the process outputs the encrypted message tagged with tag_1.

3.2 Revealing shared keys

Consider two protocols, one whose security relies on the secrecy of a shared key whereas the other protocol reveals it. Such a situation will compromise the security of the whole application. It is therefore important to ensure that shared keys are not revealed. To formalise this hypothesis, and to express the sharing of long-term keys, we introduce the notion of *composition context*. This will help us describe under which long-term keys the composition has to be done.

A *composition context* C is defined by the grammar:

$$C := \_ \mid \mathtt{new}\ n.\ C \ \mid !C \qquad \text{where } n \text{ is a name of base type.}$$

Definition 5. *Let C be a composition context, A be an extended process of the form $(\mathcal{E}; C[P]; \Phi)$, $key \in \{n, \mathsf{pk}(n), \mathsf{vk}(n) \mid n \text{ occurs in } C\}$, and c, s two fresh names. We say that A* reveals *key when*

$$(\mathcal{E} \cup \{s\}; C[P \mid \mathtt{in}(c, x).\ \mathtt{if}\ x = key\ \mathtt{then}\, \mathtt{out}(c, s)]; \Phi) \xRightarrow{\mathsf{tr}} (\mathcal{E}'; \mathcal{P}'; \Phi'; \sigma')$$

for some $\mathcal{E}'$, $\mathcal{P}'$, Φ', and σ' such that $\mathtt{new}\, \mathcal{E}'.\Phi' \vdash s$.

3.3 A first composition result

Before stating our first result regarding parallel composition for confidentiality properties, we gather the required hypotheses in the following definition.

Definition 6. *Let C be a composition context and $\mathcal{E}_0$ be a finite set of names of base type. Let P and Q be two plain processes together with their frames Φ and Ψ. We say that P/Φ and Q/Ψ are* composable *under $\mathcal{E}_0$ and C when $fv(P) = fv(Q) = \emptyset$, $\mathrm{dom}(\Phi) \cap \mathrm{dom}(\Psi) = \emptyset$, and*

1. *P (resp. Q) is built over $\Sigma_\alpha \cup \Sigma_0$ (resp. $\Sigma_\beta \cup \Sigma_0$), whereas Φ (resp. Ψ) is built over $\Sigma_\alpha \cup \{\mathsf{pk}, \mathsf{vk}, \langle\ \rangle\}$ (resp. $\Sigma_\beta \cup \{\mathsf{pk}, \mathsf{vk}, \langle\ \rangle\}$), $\Sigma_\alpha \cap \Sigma_\beta = \emptyset$, and P (resp. Q) is tagged;*
2. *$\mathcal{E}_0 \cap (fn(C[P]) \cup fn(\Phi)) \cap (fn(C[Q]) \cup fn(\Psi)) = \emptyset$; and*
3. *$(\mathcal{E}_0; C[P]; \Phi)$ (resp. $(\mathcal{E}_0; C[Q]; \Psi)$) does not reveal any key in*
$$\{n, \mathsf{pk}(n), \mathsf{vk}(n) \mid n \text{ occurs in } fn(P) \cap fn(Q) \cap bn(C)\}.$$

Condition 1 is about sharing primitives, whereas Conditions 2 and 3 ensure that keys are shared via the composition context C only (not via $\mathcal{E}_0$), and are not revealed by each protocol individually.

We are now able to state the following theorem which is in the same vein as those obtained previously in *e.g.* [15, 13]. However, the setting we consider here is more general. In particular, we consider arbitrary primitives, processes with else branches, and private channels.

Theorem 1. *Let C be a composition context, $\mathcal{E}_0$ be a finite set of names of base type, and s be a name that occurs in C. Let P and Q be two plain processes together with their frames Φ and Ψ, and assume that P/Φ and Q/Ψ are composable under $\mathcal{E}_0$ and C. If $(\mathcal{E}_0; C[P]; \Phi)$ and $(\mathcal{E}_0; C[Q]; \Psi)$ do not reveal s then $(\mathcal{E}_0; C[P \mid Q]; \Phi \uplus \Psi)$ does not reveal s.*

As most of the proofs of similar composition results, we show this result going back to the *disjoint case*. Indeed, it is well-known that parallel composition works well when protocols do not share any data (the so-called *disjoint case*). We show that all the conditions are satisfied to apply our generic result (presented only in the full version of this paper) that allows one to go back to the disjoint case. Thus, we obtain that the disjoint case $D = (\mathcal{E}_0; C[P] \mid C[Q]; \Phi \uplus \Psi)$ and the shared case $S = (\mathcal{E}_0; C[P \mid Q]; \Phi \uplus \Psi)$ are in trace equivalence, and this allows us to conclude.

4 The case of key-exchange protocols

Our goal is to go beyond parallel composition, and to further consider the particular case of key-exchange protocols. Assume that $P = \mathtt{new}\, \tilde{n}.(P_1 \mid P_2)$ is a protocol that establishes a key between two parties. The goal of P is to establish a shared session key between P_1 and P_2. Assume that P_1 stores the key in the variable x_1, while P_2 stores it in the variable x_2, and then consider a protocol Q that uses the values stored in x_1/x_2 as a fresh key to secure communications.

4.1 What is a good key exchange protocol?

In this setting, sharing between P and Q is achieved through the composition context as well as through assignment variables x_1 and x_2. The idea is to abstract these values with fresh names when we analyse Q in isolation. However, in order to abstract them in the right way, we need to know their values (or at least whether they are equal or not). This is the purpose of the property stated below.

Definition 7. *Let C be a composition context and $\mathcal{E}_0$ be a finite set of names. Let $P_1[\_]$ (resp. $P_2[\_]$) be a plain process with a hole in the scope of an assignment of the form $[x_1 := t_1]$ (resp. $[x_2 := t_2]$), and Φ be a frame.*

We say that $P_1/P_2/\Phi$ is a good *key-exchange protocol under $\mathcal{E}_0$ and C when $(\mathcal{E}_0; P_{\mathsf{good}}; \Phi)$ does not reveal bad where P_{good} is defined as follows:*

$$\begin{array}{l}
P_{\mathsf{good}} = \mathtt{new}\, bad.\mathtt{new}\, d.\big(C[\mathtt{new}\, id.(P_1[\mathtt{out}(d, \langle x_1, id\rangle)] \mid P_2[\mathtt{out}(d, \langle x_2, id\rangle)])] \\
\mid \mathtt{in}(d,x).\mathtt{in}(d,y).\mathtt{if}\ \mathsf{proj}_1(x) = \mathsf{proj}_1(y) \wedge \mathsf{proj}_2(x) \neq \mathsf{proj}_2(y)\ \mathtt{then}\ \mathtt{out}(c, bad) \\
\mid \mathtt{in}(d,x).\mathtt{in}(d,y).\mathtt{if}\ \mathsf{proj}_1(x) \neq \mathsf{proj}_1(y) \wedge \mathsf{proj}_2(x) = \mathsf{proj}_2(y)\ \mathtt{then}\ \mathtt{out}(c, bad) \\
\mid \mathtt{in}(d,x).\mathtt{in}(c,z).\mathtt{if}\ z \in \{\mathsf{proj}_1(x), \mathsf{pk}(\mathsf{proj}_1(x)), \mathsf{vk}(\mathsf{proj}_1(x))\}\ \mathtt{then}\ \mathtt{out}(c, bad)\big)
\end{array}$$

where bad is a fresh name of base type, and c, d are fresh names of channel type.

The expressions $u \neq v$ and $u \in \{v_1, \dots, v_n\}$ used above are convenient notations that can be rigorously expressed using nested conditionals. Roughly, the property expresses that x_1 and x_2 are assigned to the same value if, and only if, they are joined together, *i.e.* they share the same *id*. In particular, two instances of the role P_1 (resp. P_2) cannot assign their variable with the same value: a fresh key is established at each session. The property also ensures that the data shared through x_1/x_2 are not revealed.

Example 6. We have that $P_A/P_B/\Phi_{\mathsf{DH}}$ described in Example 2, as well as its tagged version (see Example 5) are *good* key-exchange protocols under $\mathcal{E}_0 = \{sk_A, sk_B\}$ and $C = \_$. This corresponds to a scenario where we consider only a single execution of the protocol (no replication).

Actually, the property mentioned above is quite strong, and never satisfied when the context C under study ends with a replication, *i.e.* when C is of the form $C'[!\_]$. To cope with this situation, we consider another version of this property. When C is of the form $C'[!\_]$, we define P_{good} as follows (where r_1 and r_2 are two additional fresh names of base type):

$$\begin{array}{l}
\mathtt{new}\, bad, d, r_1, r_2.\big(C'[\mathtt{new}\, id.!(P_1[\mathtt{out}(d, \langle x_1, id, r_1\rangle)] \mid P_2[\mathtt{out}(d, \langle x_2, id, r_2\rangle)])] \\
\mid \mathtt{in}(d,x).\mathtt{in}(d,y).\mathtt{if}\ \mathsf{proj}_1(x) = \mathsf{proj}_1(y) \wedge \mathsf{proj}_2(x) \neq \mathsf{proj}_2(y)\ \mathtt{then}\ \mathtt{out}(c, bad) \\
\mid \mathtt{in}(d,x).\mathtt{in}(d,y).\mathtt{if}\ \mathsf{proj}_1(x) = \mathsf{proj}_1(y) \wedge \mathsf{proj}_3(x) = \mathsf{proj}_3(y)\ \mathtt{then}\ \mathtt{out}(c, bad) \\
\mid \mathtt{in}(d,x).\mathtt{in}(c,z).\mathtt{if}\ z \in \{\mathsf{proj}_1(x), \mathsf{pk}(\mathsf{proj}_1(x)), \mathsf{vk}(\mathsf{proj}_1(x))\}\ \mathtt{then}\ \mathtt{out}(c, bad)\big)
\end{array}$$

Note that the *id* is now generated before the last replication, and thus is not uniquely associated to an instance of P_1/P_2. Instead several instances of P_1/P_2 may now share the same *id* as soon as they are identical. This gives us more flexibility. The triplet $\langle u_1, u_2, u_3\rangle$ and the operator $\mathsf{proj}_3(u)$ used above are convenient notations that can be expressed using pairs. This new version forces distinct values in the assignment variables for each instance of P_1 (resp. P_2) through the 3rd line. However, we do not fix in advance which particular instance of P_1 and P_2 should be matched, as in the first version.

Example 7. We have that $P_A/P_B/\Phi_{\mathsf{DH}}$ as well as its tagged version are good key-exchange protocols under $\mathcal{E}_0 = \{sk_A, sk_B\}$ and $C = !\,\_$.

4.2 Do we need to tag pairs?

When analysing Q in isolation, the values stored in the assignment variables x_1/x_2 are abstracted by fresh names. Since P and Q share the common signature Σ_0, we need an additional hypothesis to ensure that in any execution, the values assigned to the variables x_1/x_2 are not of the form $\langle u_1, u_2\rangle$, $\mathsf{pk}(u)$, or $\mathsf{vk}(u)$. These symbols are those of the common signature that are not tagged, thus abstracting them by fresh names in Q would not be safe. This has already been highlighted in [12]. They however left as future work the definition of the needed hypothesis and simply assume that each operator of the common signature has to be tagged. Here, we formally express the required hypothesis.

Definition 8. *An extended process A satisfies the* abstractability property *if for any $(\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ such that $A \stackrel{\mathsf{tr}}{\Longrightarrow} (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$, for any $x \in \mathrm{dom}(\sigma)$ which corresponds to an assignment variable, for any u_1, u_2, we have that $x\sigma \neq_{\mathsf{E}} \langle u_1, u_2\rangle$, $x\sigma \neq_{\mathsf{E}} \mathsf{pk}(u_1)$, and $x\sigma \neq_{\mathsf{E}} \mathsf{vk}(u_1)$.*

Note also that, in [12], the common signature is restricted to symmetric encryption and pairing only. They do not consider asymmetric encryption, and signature. Thus, our composition result generalizes theirs considering both a richer common signature, and a lighter tagging scheme (we do not tag pairs).

4.3 Composition result

We retrieve the following result which is actually a generalization of two theorems established in [12] and stated for specific composition contexts.

Theorem 2. *Let C be a composition context, $\mathcal{E}_0$ be a finite set of names of base type, and s be a name that occurs in C. Let $P_1[\_]$ (resp. $P_2[\_]$) be a plain process without replication and with an hole in the scope of an assignment of the form $[x_1 := t_1]$ (resp. $[x_2 := t_2]$). Let Q_1 (resp. Q_2) be a plain process such that $fv(Q_1) \subseteq \{x_1\}$ (resp. $fv(Q_2) \subseteq \{x_2\}$), and Φ and Ψ be two frames. Let $P = P_1[0] \mid P_2[0]$ and $Q = \mathtt{new}\ k.[x_1 := k].[x_2 := k].(Q_1 \mid Q_2)$ for some fresh name k, and assume that:*

1. *P/Φ and Q/Ψ are composable under $\mathcal{E}_0$ and C;*
2. *$(\mathcal{E}_0; C[Q]; \Psi)$ does not reveal k, $\mathsf{pk}(k)$, $\mathsf{vk}(k)$;*
3. *$(\mathcal{E}_0; C[P]; \Phi)$ satisfies the abstractability property; and*
4. *$P_1/P_2/\Phi$ is a good key-exchange protocol under $\mathcal{E}_0$ and C.*

If $(\mathcal{E}_0; C[P]; \Phi)$ *and* $(\mathcal{E}_0; C[Q]; \Psi)$ *do not reveal* s *then* $(\mathcal{E}_0; C[P_1[Q_1]|P_2[Q_2]]; \Phi \uplus \Psi)$ *does not reveal* s*.*

Basically, we prove this result relying on our generic composition result. In [12], they do not require P to be good but only ask for secrecy of the shared key. In particular they do not express any freshness or agreement property about the established key. Actually, when considering a simple composition context without replication, freshness is trivial (since there is only one session). Moreover, in their setting, agreement is not important since they do not have else branches. The analysis of Q considering that both parties have agreed on the key corresponds to the worst scenario. Note that this is not true anymore in presence of else branches. The following example shows that as soon as else branches are allowed, as it is the case in the present work, agreement becomes important.

Example 8. Consider a simple situation where:

- $P_1[0] = \mathtt{new}\, k_1.[x_1 := k_1].0$ and $P_2[0] = \mathtt{new}\, k_2.[x_2 := k_2].0$;
- $Q_1 = \mathtt{if}\ x_1 = x_2\ \mathtt{then}\ \mathtt{out}(c, \mathsf{ok})\ \mathtt{else}\ \mathtt{out}(c, s)$ and $Q_2 = 0$.

Let $\mathcal{E}_0 = \emptyset$, and $C = \mathtt{new}\ s.\_$. We consider the processes $P = P_1[0] \mid P_2[0]$, and $Q = \mathtt{new}\, k.[x_1 := k].[x_2 := k].(Q_1 \mid Q_2)$ and we assume that the frames Φ and Ψ are empty. We clearly have that $(\mathcal{E}_0; C[P]; \Phi)$ and $(\mathcal{E}_0; C[Q]; \Psi)$ do not reveal s whereas $(\mathcal{E}_0; C[P_1[Q_1] \mid P_2[Q_2]; \Phi \uplus \Psi)$ does. The only hypothesis of Theorem 2 that is violated is the fact that $P_1/P_2/\Phi$ is not a good key-exchange protocol due to a lack of agreement on the key which is generated (*bad* can be emitted thanks to the 3rd line of the process P_{good} given in Definition 7).

Now, regarding their second theorem corresponding to a context of the form $\mathtt{new}\, s.\,!\_$, as before agreement is not mandatory but freshness of the key established by the protocol P is crucial. As illustrated by the following example, this hypothesis is missing in the theorem stated in [12] (Theorem 3).

Example 9. Consider $A = (\{k_P\}; \mathtt{new}\, s.!([x_1 := k_P].0 \mid [x_2 := k_P].0); \emptyset)$, as well as $B = (\{k_P\}; \mathtt{new}\, s.\,!Q; \emptyset)$ where $Q = \mathtt{new}\, k.[x_1 := k].[x_2 := k].(Q_1 \mid Q_2)$ with

$$Q_1 = \mathtt{out}(c, \mathsf{senc}(\mathsf{senc}(s, k), k));\ \text{and}\ Q_2 = \mathtt{in}(c, x).\mathtt{out}(c, \mathsf{sdec}(x, k)).$$

Note that neither A nor B reveals s. In particular, the process Q_1 emits the secret s encrypted twice with a fresh key k, but Q_2 only allows us to remove one level of encryption with k. Now, if we plug the key-exchange protocol given above with no guarantee of freshness (the same key is established at each session), the resulting process, *i.e.* $(\mathcal{E}_0; C[P_1[Q_1] \mid P_2[Q_2]]; \emptyset)$ does reveal s.

Note that this example is not a counter example of our Theorem 2: $P_1/P_2/\emptyset$ is not a good key-exchange protocol according to our definition.

5 Dealing with equivalence-based properties

Our ultimate goal is to analyse privacy-type properties in a modular way. In [4], we propose several composition results w.r.t. privacy-type properties, but for parallel composition only. Here, we want to go beyond parallel composition, and consider the case of key-exchange protocols.

5.1 A problematic example

Even in a quite simple setting (the shared keys are not revealed, protocols do not share any primitives), such a sequential composition result does not hold. Let $C = \mathtt{new}\ k.!\,\mathtt{new}\ k_1.!\,\mathtt{new}\ k_2.\ \_$ be a composition context, yes/no, ok/ko be public constants, $u = \mathsf{senc}(\langle k_1, k_2\rangle, k)$, and consider the following processes:

$$\begin{aligned} Q(z_1, z_2) \;&=\; \mathtt{out}(c, u).\mathtt{in}(c, x).\mathtt{if}\ x = u\ \mathtt{then}\ 0\ \mathtt{else} \\ &\;\ \mathtt{if}\ \mathsf{proj}_1(\mathsf{sdec}(x, k)) = k_1\ \mathtt{then}\ \mathtt{out}(c, z_1)\ \mathtt{else}\ \mathtt{out}(c, z_2) \\ P[\_] \;&=\; \mathtt{out}(c, u).\big(\_ \mid \mathtt{in}(c, x).\mathtt{if}\ x = u\ \mathtt{then}\ 0\ \mathtt{else} \\ &\;\ \mathtt{if}\ \mathsf{proj}_1(\mathsf{sdec}(x, k)) = k_1\ \mathtt{then}\ \mathtt{out}(c, \mathsf{ok})\ \mathtt{else}\ \mathtt{out}(c, \mathsf{ko})\big) \end{aligned}$$

We have that $C[P[0]] \approx C[P[0]]$ and also that $C[Q(\mathsf{yes},\mathsf{no})] \approx C[Q(\mathsf{no},\mathsf{yes})]$. This latter equivalence is non-trivial. Intuitively, when $C[Q(\mathsf{yes},\mathsf{no})]$ unfolds its outermost ! and then performs an output, then $C[Q(\mathsf{no},\mathsf{yes})]$ has to mimic this step by unfolding its innermost ! and by performing the only available output. This will allow it to react in the same way as $C[Q(\mathsf{yes},\mathsf{no})]$ in case encrypted messages are used to fill some input actions. Since the two processes $P[0]$ and $Q(\mathsf{yes},\mathsf{no})$ (resp. $Q(\mathsf{no},\mathsf{yes})$) are almost "disjoint", we could expect the equivalence $C[P[Q(\mathsf{yes},\mathsf{no})]] \approx C[P[Q(\mathsf{no},\mathsf{yes})]]$ to hold. Actually, this equivalence does *not* hold. The presence of the process P gives to the attacker some additional distinguishing power. In particular, through the outputs ok/ko outputted by P, the attacker will learn which ! has been unfolded. This result holds even if we rename function symbols so that protocols P and Q do not share any primitives. The problem is that the two equivalences we want to compose hold for different reasons, *i.e.* by unfolding the replications in a different and incompatible way. Thus, when the composed process $C[P[Q(\mathsf{yes},\mathsf{no})]]$ reaches a point where $Q(\mathsf{yes},\mathsf{no})$ can be executed, on the other side, the process $Q(\mathsf{no},\mathsf{yes})$ is ready to be executed but the instance that is available is not the one that was used when establishing the equivalence $C[Q(\mathsf{yes},\mathsf{no})] \approx C[Q(\mathsf{no},\mathsf{yes})]$. Therefore, in order to establish equivalence-based properties in a modular way, we rely on a stronger notion of equivalence, namely *diff-equivalence*, that will ensure that the two "small" equivalences are satisfied in a compatible way.

Note that this problem does not arise when considering reachability properties and/or parallel composition. In particular, we have that:

$$C[P[0] \mid Q(\mathsf{yes},\mathsf{no})] \approx C[P[0] \mid Q(\mathsf{no},\mathsf{yes})].$$

5.2 Biprocesses and diff-equivalence

We consider pairs of processes, called *biprocesses*, that have the same structure and differ only in the terms and tests that they contain. Following the approach of [9], we introduce a special symbol diff of arity 2 in our signature. The idea being to use this diff operator to indicate when the terms manipulated by the processes are different. Given a biprocess B, we define two processes $\mathsf{fst}(B)$ and $\mathsf{snd}(B)$ as follows: $\mathsf{fst}(B)$ is obtained by replacing each occurrence of $\mathsf{diff}(M, M')$ (resp. $\mathsf{diff}(\varphi, \varphi')$) with M (resp. φ), and similarly $\mathsf{snd}(B)$ is obtained by replacing each occurrence of $\mathsf{diff}(M, M')$ (resp. $\mathsf{diff}(\varphi, \varphi')$) with M' (resp. φ').

The semantics of biprocesses is defined as expected via a relation that expresses when and how a biprocess may evolve. A biprocess reduces if, and only if, both sides of the biprocess reduce in the same way: a communication succeeds on both sides, a conditional has to be evaluated in the same way in both sides too. For instance, the `then` and `else` rules are as follows:

$$(\mathcal{E}; \{\mathtt{if}\ \mathsf{diff}(\varphi_L, \varphi_R)\ \mathtt{then}\ Q_1\ \mathtt{else}\ Q_2\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau}_{\mathsf{bi}} (\mathcal{E}; Q_1 \uplus \mathcal{P}; \Phi; \sigma)$$
$$\text{if } u\sigma =_{\mathsf{E}} v\sigma \text{ for each } u = v \in \varphi_L, \text{ and } u'\sigma =_{\mathsf{E}} v'\sigma \text{ for each } u' = v' \in \varphi_R$$

$$(\mathcal{E}; \{\mathtt{if}\ \mathsf{diff}(\varphi_L, \varphi_R)\ \mathtt{then}\ Q_1\ \mathtt{else}\ Q_2\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau}_{\mathsf{bi}} (\mathcal{E}; Q_2 \uplus \mathcal{P}; \Phi; \sigma)$$
$$\text{if } u\sigma \neq_{\mathsf{E}} v\sigma \text{ for some } u = v \in \varphi_L, \text{ and } u'\sigma \neq_{\mathsf{E}} v'\sigma \text{ for some } u' = v' \in \varphi_R$$

When the two sides of the biprocess reduce in different ways, the biprocess blocks. The relation $\xRightarrow{\mathsf{tr}}_{\mathsf{bi}}$ on biprocesses is defined as for processes. This leads us to the following notion of *diff-equivalence*.

Definition 9. *An extended biprocess B_0 satisfies* diff-equivalence *if for every biprocess $B = (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ such that $B_0 \xRightarrow{\mathsf{tr}}_{\mathsf{bi}} B$ for some trace* tr*, we have that*

1. $\mathit{new}\ \mathcal{E}.\mathsf{fst}(\Phi) \sim \mathit{new}\ \mathcal{E}.\mathsf{snd}(\Phi)$
2. *if $\mathsf{fst}(B) \xrightarrow{\ell} A_L$ then there exists B' such that $B \xrightarrow{\ell}_{\mathsf{bi}} B'$ and $\mathsf{fst}(B') = A_L$ (and similarly for* snd*).*

The notions introduced so far on processes are extended as expected on biprocesses: the property has to hold on both $\mathsf{fst}(B)$ and $\mathsf{snd}(B)$. Sometimes, we also say that the biprocess B is in trace equivalence instead of writing $\mathsf{fst}(B) \approx \mathsf{snd}(B)$.

As expected, this notion of diff-equivalence is actually stronger than the usual notion of trace equivalence.

Lemma 1. *A biprocess B that satisfies diff-equivalence is in trace equivalence.*

6 Composition results for diff-equivalence

We first consider the case of parallel composition. This result is in the spirit of the one established in [4]. However, we adapt it to diff-equivalence in order to combine it with the composition result we obtained for the the case of key-exchange protocol (see Theorem 4).

Theorem 3. *Let C be a composition context and $\mathcal{E}_0$ be a finite set of names of base type. Let P and Q be two plain biprocesses together with their frames Φ and Ψ, and assume that P/Φ and Q/Ψ are composable under $\mathcal{E}_0$ and C.*

If $(\mathcal{E}_0; C[P]; \Phi)$ and $(\mathcal{E}_0; C[Q]; \Psi)$ satisfy diff-equivalence (resp. trace equivalence) then the biprocess $(\mathcal{E}_0; C[P \mid Q]; \Phi \uplus \Psi)$ satisfies diff-equivalence (resp. trace equivalence).

Proof. (sketch) As for the proof for Theorem 1, parallel composition works well when processes do not share any data. Hence, we easily deduce that $D = (\mathcal{E}_0; C[P] \mid C[Q]; \Phi \uplus \Psi)$ satisfies the diff-equivalence (resp. trace equivalence).

Then, we compare the behaviours of the biprocess D to those of the biprocess $S = (\mathcal{E}_0; C[P \mid Q]; \Phi \uplus \Psi)$. More precisely, this allows us to establish that $\mathsf{fst}(D)$ and $\mathsf{fst}(S)$ are in diff-equivalence (as well as $\mathsf{snd}(D)$ and $\mathsf{snd}(S)$), and then we conclude relying on the transitivity of the equivalence. $\square$

Now, regarding sequential composition and the particular case of key-exchange protocols, we obtain the following composition result.

Theorem 4. *Let C be a composition context and $\mathcal{E}_0$ be a finite set of names of base type. Let $P_1[\_]$ (resp. $P_2[\_]$) be a plain biprocess without replication and with an hole in the scope of an assignment of the form $[x_1 := t_1]$ (resp. $[x_2 := t_2]$). Let Q_1 (resp. Q_2) be a plain biprocess such that $fv(Q_1) \subseteq \{x_1\}$ (resp. $fv(Q_2) \subseteq \{x_2\}$), and Φ and Ψ be two frames. Let $P = P_1[0] \mid P_2[0]$ and $Q = \texttt{new}\ k.[x_1 := k].[x_2 := k].(Q_1 \mid Q_2)$ for some fresh name k, and assume that:*

1. *P/Φ and Q/Ψ are composable under $\mathcal{E}_0$ and C;*
2. *$(\mathcal{E}_0; C[Q]; \Psi)$ does not reveal k, $\mathsf{pk}(k)$, $\mathsf{vk}(k)$;*
3. *$(\mathcal{E}_0; C[P]; \Phi)$ satisfies the abstractability property; and*
4. *$P_1/P_2/\Phi$ is a good key-exchange protocol under $\mathcal{E}_0$ and C.*

Let $P^+ {=} P_1[\texttt{out}(d, x_1)] \mid P_2[\texttt{out}(d, x_2)] \mid \texttt{in}(d, x).\texttt{in}(d, y).\texttt{if}\ x = y\ \texttt{then}\ 0\ \texttt{else}\ 0$. If the biprocesses $(\mathcal{E}_0; \texttt{new}\ d.C[P^+]; \Phi)$ and $(\mathcal{E}_0; C[Q]; \Psi)$ satisfy diff-equivalence then $(\mathcal{E}_0; C[P_1[Q_1] \mid P_2[Q_2]]; \Phi \uplus \Psi)$ satisfies diff-equivalence.

We require $(\mathcal{E}_0; \texttt{new}\ d.C[P^+]; \Phi)$ to be in diff-equivalence (and not simply $(\mathcal{E}_0; C[P]; \Phi)$). This ensures that the same equalities between values of assignment variables hold on both sides of the equivalence. Actually, when the composition context C under study is not of the form $C'[!\_]$, and under the hypothesis that $P_1/P_2/\Phi$ is a good key-exchange protocol under $\mathcal{E}_0$ and C, we have that these two requirements coincide. However, the stronger hypothesis is important to conclude when C is of the form $C'[!\_]$. Indeed, in this case, we do not know in advance what are the instances of P_1 and P_2 that will be "matched". This is not a problem but to conclude about the diff-equivalence of the whole process (*i.e.* $(\mathcal{E}_0; C[P_1[Q_1] \mid P_2[Q_2]]; \Phi \uplus \Psi)$), we need to ensure that such a matching is the same on both sides of the equivalence. Note that to conclude about trace equivalence only, this additional requirement is actually not necessary.

7 Case studies

Many applications rely on several protocols running in composition (parallel, sequential, or nested). In this section, we show that our results can help in the analysis of this sort of complex system. Our main goal is to show that the extra hypotheses needed to analyse an application in a moduar way are reasonnable.

7.1 3G mobile phones

We look at confidentiality and privacy guarantees provided by the *AKA* protocol and the Submit SMS procedure (*sSMS*) when run in composition as specified by the 3GPP consortium in [2].

Protocols description. The *sSMS* protocol allows a mobile station (MS) to send an SMS to another MS through a serving network (SN). The confidentiality of the sent SMS relies on a session key *ck* established through the execution of the *AKA* protocol between the MS and the SN. The *AKA* protocol achieves mutual authentication between a MS and a SN, and allows them to establish a shared session key *ck*. The *AKA* protocol consists in the exchange of two messages: the *authentication request* and the *authentication response*. The *AKA* protocol as deployed in real 3G telecommunication systems presents a linkability attack [5], and thus we consider here its fixed version as described in [5]. At the end of a successful execution of this protocol, both parties should agree on a fresh ciphering key *ck*. This situation can be modelled in our calculus as follows:

$$\texttt{new}\ sk_{SN}.\ !\texttt{new}\ IMSI.\ \texttt{new}\ k_{IMSI}.\ !\texttt{new}\ sqn.\ \texttt{new}\ sms.\\ (AKA^{SN}[sSMS^{SN}] \mid AKA^{MS}[sSMS^{MS}])$$

where sk_{SN} represents the private key of the network; while *IMSI* and k_{IMSI} represent respectively the long-term identity and the symmetric key of the MS. The name *sqn* models the sequence number on which SN and MS are synchronised. The two subprocesses AKA^{MS} and $sSMS^{MS}$ (*resp.* AKA^{SN}, and $sSMS^{SN}$) model one session of the MS's (*resp.* SN's) side of the *AKA*, and *sSMS* protocols respectively. Each MS, identified by its identity *IMSI* and its key k_{IMSI}, can run multiple times the *AKA* protocol followed by the *sSMS* protocol.

Security analysis. We explain how some confidentiality and privacy properties of the *AKA* protocol and the *sSMS* procedure can be derived relying on our composition results. We do not need to tag the protocols under study to perform our analysis since they do not share any primitive but the pairing operator. Note that the *AKA* protocol can *not* be modelled in the calculus given in [12] due to the need of non-trivial else branches. Moreover, to enable the use of ProVerif, we had to abstract some details of the considered protocols that ProVerif cannot handle. In particular, we model timestamps using nonces, we replace the use of the xor operation by symmetric encryption, and we assume that the two parties are "magically" synchronised on their counter value.

Strong unlinkability requires that an observer does not see the difference between the two following scenarios: *(i)* a same mobile phone sends several SMSs; or *(ii)* multiple mobile phones send at most one SMS each. To model this requirement, we consider the composition context[5]:

$$\begin{aligned} C_U[\_] \overset{\mathsf{def}}{=}\ & !\texttt{new}\ IMSI_1.\ \texttt{new}\ k_{IMSI_1}.\ !\texttt{new}\ IMSI_2.\ \texttt{new}\ k_{IMSI_2}.\\ & \texttt{let}\ IMSI = \mathsf{diff}[IMSI_1, IMSI_2]\ \texttt{in}\ \texttt{let}\ k_{IMSI} = \mathsf{diff}[k_{IMSI_1}, k_{IMSI_2}]\ \texttt{in}\\ & \texttt{new}\ sqn.\ \texttt{new}\ sms.\ \_ \end{aligned}$$

To check if the considered 3G protocols satisfy strong unlinkability, one needs to check if the following biprocess satisfies diff-equivalence ($\Phi_0 = \{w_1 \rhd \mathsf{pk}(sk_{SN})\}$):

$$(sk_{SN}; C_U[AKA^{SN}[sSMS^{SN}] \mid AKA^{MS}[sSMS^{MS}]]; \Phi_0)$$

[5] We use $\texttt{let}\ x = M\ \texttt{in}\ P$ to denote the process $P\{M/x\}$.

Hypotheses (1-4) stated in Theorem 4 are satisfied, and thus this equivalence can be derived from the following two "smaller" diff-equivalences:

$$(sk_{SN}; \texttt{new}\ d.\ C_U[AKA^+]; \Phi_0) \quad \text{and} \quad (sk_{SN}; C'_U[sSMS]; \emptyset)$$

- $sSMS \stackrel{\mathsf{def}}{=} sSMS^{SN} \mid sSMS^{MS}$,
- $AKA^+ \stackrel{\mathsf{def}}{=} AKA^{SN}[\texttt{out}(d, xck_{SN})] \mid AKA^{MS}[\texttt{out}(d, xck_{MS})] \mid$
  $\texttt{in}(d, x).\ \texttt{in}(d, y).\ \texttt{if}\ x = y\ \texttt{then}\ 0\ \texttt{else}\ 0$
- $C'_U[\_] \stackrel{\mathsf{def}}{=} C_U[\texttt{new}\ ck.\texttt{let}\ xck_{SN} = ck\ \texttt{in}\ \texttt{let}\ xck_{MS} = ck\ \texttt{in}\ \_]$.

Weak secrecy requires that the sent/received SMS is not deducible by an outsider, and can be modelled using the context

$$C_{WS}[\_] \stackrel{\mathsf{def}}{=} !\texttt{new}\ IMSI.\ \texttt{new}\ k_{IMSI}.\ !\texttt{new}\ sqn.\texttt{new}\ sms.\_.$$

The composition context C_{WS} is the same as $\mathsf{fst}(C_U)$ (up to some renaming), thus Hypotheses (1-4) of Theorem 2 also hold and we derive the weak secrecy property by simply analysing this property on AKA and $sSMS$ in isolation.

Strong secrecy means that an outsider should not be able to distinguish the situation where sms_1 is sent (resp. received), from the situation where sms_2 is sent (resp. received), although he might know the content of sms_1 and sms_2. This can be modelled using the following composition context:

$$C_{SS}[\_] \stackrel{\mathsf{def}}{=} !\texttt{new}\ IMSI.\ \texttt{new}\ k_{IMSI}.\ !\texttt{new}\ sqn.\ \texttt{let}\ sms = \mathsf{diff}[sms_1, sms_2]\ \texttt{in}\ \_$$

where sms_1 and sms_2 are two free names known to the attacker. Again, our Theorem 4 allows us to reason about this property in a modular way.

Under the abstractions briefly explained above, all the hypotheses have been checked using ProVerif. Actually, it happens that ProVerif is also able to conclude on the orignal protocol (the one without decomposition) for the three security properties mentioned above. Note that a less abstract model of the same protocol (*e.g.* the one with the xor operator) would have required us to rely on a manual proof. In such a situation, our composition result allows us to reduce a big equivalence that existing tools cannot handle, to a much smaller one which is a more manageable work in case the proof has to be done manually.

7.2 E-passport application

We look at privacy guarantees provided by three protocols of the e-passport application when run in composition as specified in [1].

Protocols description. The information stored in the chip of the passport is organised in data groups (dg_1 to dg_{19}): dg_5 contains a JPEG copy of the displayed picture, dg_7 contains the displayed signature, whereas the verification key $\mathsf{vk}(sk_P)$ of the passport, together with its certificate $\mathsf{sign}(\mathsf{vk}(sk_P), sk_{DS})$ issued by the Document Signer authority are stored in dg_{15}. For authentication purposes, a hash of all the dgs together with a signature on this hash value are stored in a separate file, the Security Object Document:

$$sod \stackrel{\mathsf{def}}{=} \langle \mathsf{sign}(\mathsf{h}(dg_1, \ldots, dg_{19}), sk_{DS}),\ \mathsf{h}(dg_1, \ldots, dg_{19})\rangle.$$

The ICAO standard specifies several protocols through which this information can be accessed [1]. First, the *Basic Access Control* (*BAC*) protocol establishes a key seed *kseed* from which a session key *kenc* is derived. The purpose of *kenc* is to prevent skimming and eavesdropping on the subsequent communication with the e-passport. The security of the *BAC* protocol relies on two master keys, *ke* and *km*. Once the *BAC* protocol has been successfully executed, the reader gains access to the information stored in the RFID tag through the *Passive Authentication* (*PA*) and the *Active Authentication* (*AA*) protocols that can be executed in any order. This situation can be modelled in our calculus:

$$P \stackrel{\mathsf{def}}{=} \mathtt{new}\ sk_{DS}.\ !\mathtt{new}\ ke.\ \mathtt{new}\ km.\ \mathtt{new}\ sk_P.\mathtt{new}\ id.\ \mathtt{new}\ sig.\ \mathtt{new}\ pic.\ \dots \\ !(BAC^R[PA^R \mid AA^R] \mid BAC^P[PA^P \mid AA^P])$$

where id, sig, pic, ... represent the name, the signature, the displayed picture, *etc* of the e-passport's owner, *i.e.* the data stored in the *dg*s (1-14) and (16-19). The subprocesses BAC^P, PA^P and AA^P (*resp.* BAC^R, PA^R and AA^R) model one session of the passport's (*resp.* reader's) side of the *BAC*, *PA* and *AA* protocols respectively. The name sk_{DS} models the signing key of the Document Signing authority used in all passports. Each passport (identified by its master keys *ke* and *km*, its signing key sk_P, the owner's name, picture, signature, ...) can run multiple times the *BAC* protocol followed by the *PA* and *AA* protocols.

Security analysis. We explain below how *strong anonymity* of these three protocols executed together can be derived from the analysis performed on each protocol in isolation. In [4], as sequential composition could not be handled, the analysis of the e-passports application had to exclude the execution of the *BAC* protocol. Instead, it was assumed that the key *kenc* is "magically" pre-shared between the passport and the reader. Thanks to our Theorem 4, we are now able to complete the analysis of the e-passport application.

To express strong anonymity, we need on the one hand to consider a system in which the particular e-passport with publicly known id_1, sig_1, pic_1, *etc.* is being executed, while on the other hand it is a different e-passport with publicly known id_2, sig_2, pic_2, *etc.* which is being executed. We consider the context:

$$C_A[\_] \stackrel{\mathsf{def}}{=} !\mathtt{new}\ ke.\ \mathtt{new}\ km.\ \mathtt{new}\ sk_P.\mathtt{let}\ id = \mathsf{diff}[id_1, id_2]\ \mathtt{in}\ \ \dots !\ \_$$

This composition context differs in the e-passport being executed on the left-hand process and on the right-hand process. In other words, the system satisfies anonymity if an observer cannot distinguish the situation where the e-passport with publicly known id_1, sig_1, pic_1, *etc.* is being executed, from the situation where it is another e-passport which is being executed. To check if the tagged version of the e-passport application (we assume here that *BAC*, *PA*, and *AA* are tagged in different ways) preserves strong anonymity, one thus needs to check if the following biprocess satisfies diff-equivalence (with $\Phi_0 = \{w_1 \triangleright \mathsf{vk}(sk_{DS})\}$):

$$(sk_{DS}; C_A[BAC^R[PA^R \mid AA^R] \mid BAC^P[PA^P \mid AA^P]]; \Phi_0)$$

We can instead check whether *BAC*, *PA* and *AA* satisfy anonymity in isolation, *i.e.* if the following three diff-equivalences hold:

$$(sk_{DS}; \texttt{new}\ d.\ C_A[BAC^{+}]; \emptyset)\ (\alpha) \qquad \begin{array}{c}(sk_{DS}; C'_A[PA^R \mid PA^P]; \Phi_0)\ (\beta)\\ (sk_{DS}; C'_A[AA^R \mid AA^P]; \emptyset)\ (\gamma)\end{array}$$

where

- $BAC^{+} \stackrel{\mathsf{def}}{=} \begin{array}{l} BAC^R[\texttt{out}(d, xkenc_R)] \ \mid\ BAC^P[\texttt{out}(d, xkenc_P)] \\ \mid \texttt{in}(d,x).\ \texttt{in}(d,y).\ \texttt{if}\ x = y\ \texttt{then}\ 0\ \texttt{else}\ 0; \end{array}$
- $C'_A[\_] \stackrel{\mathsf{def}}{=} C_A[C''_A[\_]]$; and
- $C''_A[\_] \stackrel{\mathsf{def}}{=} \texttt{new}\ kenc.\ \texttt{let}\ xkenc_R = kenc\ \texttt{in}\ \texttt{let}\ xkenc_P = kenc\ \texttt{in}\ \_$.

Then, applying Theorem 3 to (β) and (γ) we derive that the following biprocess satisfies diff-equivalence:

$$(sk_{DS}; C'_A[PA^R \mid AA^R \mid PA^P \mid AA^P]; \Phi_0) \quad (\delta).$$

and applying Theorem 4 to (α) and (δ), we derive the required diff-equivalence:

$$(sk_{DS}; C_A[BAC^R[PA^R \mid AA^R] \mid BAC^P[PA^P \mid AA^P]]; \Phi_0)$$

Note that we can do so because Hypotheses (1-4) stated in Theorem 4 are satisfied, and in particular because $BAC^R/BAC^P/\emptyset$ is a good key-exchange protocol under $\{sk_{DS}\}$ and C_A. Again, all the hypotheses have been checked using ProVerif. Actually, it happens that ProVerif is also able to directly conclude on the whole system.

Unfortunately, our approach does not apply to perform a modular analysis of strong unlinkability. The BAC protocol does not satisfy the diff-equivalence needed to express such a security property, and this hypothesis is mandatory to apply our composition result.

8 Conclusion

We investigate composition results for reachability properties as well as privacy-type properties expressed using a notion of equivalence. Relying on a generic composition result, we derive parallel composition results, and we study the particular case of key-exchange protocols under various composition contexts.

All these results work in a quite general setting, *e.g.* processes may have non trivial else branches, we consider arbitrary primitives expressed using an equational theory, and processes may even share some standard primitives as long as they are tagged in different ways. We illustrate the usefulness of our results through the mobile phone and e-passport applications.

We believe that our generic result could be used to derive further composition results. We may want for instance to relax the notion of being a *good protocol* at the price of studying a less ideal scenario when analysing the protocol Q in isolation. We may also want to consider situations where sub-protocols sharing some data are arbitrarily interleaved. Moreover, even if we consider arbitrary primitives, sub-protocols can only share some standard primitives provided that they are tagged. It would be nice to relax these conditions. This would allow one to compose protocols (and not their tagged versions) or to compose protocols that both rely on primitives for which no tagging scheme actually exists (*e.g.* exclusive-or).

References

1. PKI for machine readable travel documents offering ICC read-only access. Technical report, International Civil Aviation Organization, 2004.
2. 3GPP. Technical specification group services and system aspects; 3G security; security architecture (release 9). Technical report, 3rd Generation Partnership Project, 2010.
3. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages (POPL'01)*, 2001.
4. M. Arapinis, V. Cheval, and S. Delaune. Verifying privacy-type properties in a modular way. In *Proc. 25th IEEE Computer Security Foundations Symposium (CSF'12)*, 2012.
5. M. Arapinis, L. I. Mancini, E. Ritter, M. Ryan, N. Golde, K. Redon, and R. Borgaonkar. New privacy issues in mobile telephony: fix and verification. In *ACM Conference on Computer and Communications Security*, 2012.
6. A. Armando, R. Carbone, L. Compagna, J. Cuéllar, and M. L. Tobarra. Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps. In *Proc. 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*, 2008.
7. A. Armando et al. The AVANTSSAR Platform for the Automated Validation of Trust and Security of Service-Oriented Architectures. In *Proc. 18th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2012.
8. B. Barak, R. Canetti, J. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *Proc. 45th Symposium on Foundations of Computer Science (FOCS'04)*, 2004.
9. B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 2008.
10. F. Böhl and D. Unruh. Symbolic universal composability. In *Proc. 26th Computer Security Foundations Symposium (CSF'13)*, 2013.
11. M. Bruso, K. Chatzikokolakis, and J. den Hartog. Formal verification of privacy for RFID systems. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*, 2010.
12. Ş. Ciobâcă and V. Cortier. Protocol composition for arbitrary primitives. In *Proc. of the 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, 2010.
13. V. Cortier and S. Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1):1–36, Feb. 2009.
14. T. Groß and S. Mödersheim. Vertical protocol composition. In *Proc. 24th Computer Security Foundations Symposium, (CSF'11)*, 2011.
15. J. D. Guttman and F. J. Thayer. Protocol independence through disjoint encryption. In *Proc. 13th Computer Security Foundations Workshop (CSFW'00)*, 2000.
16. R. Küsters and M. Tuengerthal. Composition Theorems Without Pre-Established Session Identifiers. In *Proc. 18th Conference on Computer and Communications Security (CCS'11)*, 2011.
17. S. Mödersheim and L. Viganò. Secure pseudonymous channels. In *Proc. 14th European Symposium on Research in Computer Security (ESORICS'09)*, 2009.
18. A. Tiu and J. E. Dawson. Automating open bisimulation checking for the spi calculus. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*, 2010.

A Case study: 3G mobile phones

In this section, we look at the confidentiality and privacy guarantees provided by the Authentication and Key Agreement protocol (*AKA*) and the Submit SMS procedure (*sSMS*), when run in composition as specified by the 3GPP consortium in [2].

The *AKA* protocol achieves mutual authentication between a Mobile Station (MS) and the Serving Network (SN), and allows them to establish shared session keys to be used to secure subsequent communications. We consider here its fixed version as described in [5] which relies on a public key infrastructure. In particular, in case of failure, *i.e.* φ_{test} is not satisfied, the answer RES is encrypted using the public key of the SN, *i.e.* $\mathsf{pk}(sk_{SN})$.

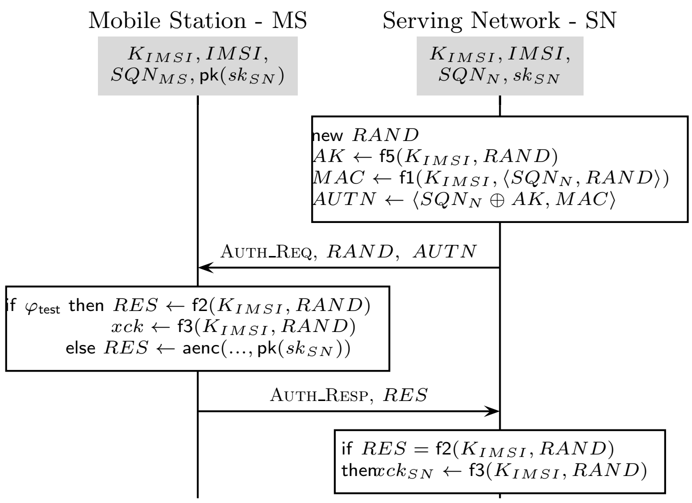
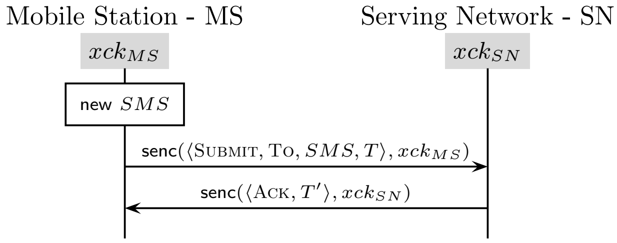


Fig. 2. The *AKA* protocol (variant proposed in [5])

The functions $\mathsf{f1} - \mathsf{f5}$, used to compute the authentication parameters, are one-way keyed cryptographic functions, and $\oplus$ denotes the exclusive-or operator. $AUTN$ contains a MAC of the concatenation of the random number with a sequence number SQN_N generated by the network using an individual counter for each subscriber. The sequence number SQN_N allows the mobile station to verify the freshness of the authentication request to defend against replay attacks. The mobile station computes the ciphering key ck and stores it in xck_{MS}. It also computes the authentication response RES and sends it to the network. The network authenticates the mobile station by verifying whether the received response is equal to the expected one. If so, the network also computes its version of the key ck and stores it in xck_{SN}.

The *sSMS* protocol allows a MS to send an SMS to another MS through the Network. The confidentiality of the sent SMS relies on the session key ck

established through the execution of the AKA protocol between the MS and the network.



It is always the MS that initiates the *sSMS* procedure. It does so by encrypting the content of the SMS it wants to submit, together with the number of the destination MS and a timestamp T, with the session key *ck* previously established. The message also contains a constant SUBMIT. The Network acknowledges the receipt of this message with a message that includes a constant ACK and a timestamp T', encrypted with *ck*.

Security analysis. The *sSMS* procedure uses a ciphering session key CK established through the execution of the AKA protocol for the confidentiality of the sent and received SMSs. We can thus use Theorem 2 and Theorem 4 to reason in a modular way about the confidentiality and privacy guarantees provided by these two protocols.

Strong unlinkability requires that an outside observer does not see the difference between the two following scenarios: *(i)* a same mobile phone sends several SMSs; or *(ii)* multiple mobile phones send at most one SMS each. To model this requirement, we consider the composition context[6]:

$$\begin{aligned} C_U[\_] \stackrel{\mathsf{def}}{=} \; & !\mathtt{new}\ IMSI_1.\ \mathtt{new}\ k_{IMSI\,1}.\ !\mathtt{new}\ IMSI_2.\ \mathtt{new}\ k_{IMSI\,2}. \\ & \mathtt{let}\ IMSI = \mathsf{diff}[IMSI_1, IMSI_2]\ \mathtt{in}\ \mathtt{let}\ k_{IMSI} = \mathsf{diff}[k_{IMSI\,1}, k_{IMSI\,2}]\ \mathtt{in} \\ & \mathtt{new}\ sqn.\ \mathtt{new}\ sms.\ \_ \end{aligned}$$

In the left-hand process, the identity of the phone in the filling process is $IMSI_1$ and the long-term key is $k_{IMSI\,1}$, allowing the same phone to execute multiple times the AKA protocol followed by the *sSMS* protocol. In the right-hand process, the values that are used in the filling process are $IMSI_2$ and $k_{IMSI\,2}$, restricting the execution of the considered protocols to at most one time. To check if the considered 3G protocols satisfy strong unlinkability, one needs to check if the following biprocess satisfies diff-equivalence:

$$(sk_{SN}; C_U[AKA^{SN}[sSMS^{SN}] \mid AKA^{MS}[sSMS^{MS}]]; \Phi_0) \quad \text{where } \Phi_0 = \{w_1 \rhd \mathsf{pk}(sk_{SN})\}.$$

Actually, thanks Theorem 4, this equivalence can be derived from the following two smaller diff-equivalences:

$$(sk_{SN}; \mathtt{new}\ d.\ C_U[AKA^{+}]; \Phi_0) \quad \text{and} \quad (sk_{SN}; C'_U[sSMS]; \emptyset)$$

[6] We use $\mathtt{let}\ x = M\ \mathtt{in}\ P$ to denote the process $P\{M/x\}$.

where $sSMS \stackrel{\mathsf{def}}{=} sSMS^{SN} \mid sSMS^{MS}$,

$$\begin{aligned} AKA^{+} \stackrel{\mathsf{def}}{=} \; & AKA^{SN}[\mathtt{out}(d, xck_{SN})] \mid AKA^{MS}[\mathtt{out}(d, xck_{MS})] \mid \\ & \mathtt{in}(d, x).\ \mathtt{in}(d, y).\ \mathtt{if}\ x = y\ \mathtt{then}\ 0\ \mathtt{else}\ 0 \end{aligned}$$

and $C'_U[\_] \stackrel{\mathsf{def}}{=} C_U[\mathtt{new}\ ck.\mathtt{let}\ xck_{SN} = ck\ \mathtt{in}\ \mathtt{let}\ xck_{MS} = ck\ \mathtt{in}\ \_]$.

Indeed, let $P = AKA^{SN}[0] \mid AKA^{MS}[0]$ and $Q = \mathtt{new}\ ck.[xck_{SN} := ck].[xck_{MS} := ck].(sSMS^{SN} \mid sSMS^{MS})$, and assume that Ψ is the empty frame. Considering the AKA and $sSMS$ protocols, we can check that P/Φ_0 and Q/Ψ are composable under $\mathcal{E}_0 = \{sk_{SN}\}$ and C_U (according to Definition 6). Note that $fn(P) \cap fn(Q) \cap bn(C_U) = \emptyset$, and thus the last condition trivially holds. Furthermore, using ProVerif we can show that the remaining properties are also satisfied, and that the two "small" equivalences also hold.

Weak secrecy requires that the sent/received SMS is not deducible by an outsider, and can be modelled using the context

$$C_{WS}[\_] \stackrel{\mathsf{def}}{=} !\mathtt{new}\ IMSI.\ \mathtt{new}\ k_{IMSI}.\ !\mathtt{new}\ sqn.\mathtt{new}\ sms.\_.$$

To check if the considered 3G protocols satisfy weak secrecy of sent/received SMSs *w.r.t.* some initial intruder knowledge, *e.g.* $\Phi_0 = \{w_1 \rhd \mathsf{pk}(sk_{SN})\}$, one needs to check if the following process does not reveal sms

$$(sk_{SN}; C_{WS}[AKA^{SN}[sSMS^{SN}] \mid AKA^{MS}[sSMS^{MS}]]; \Phi_0).$$

However, according to Theorem 2 we can instead check whether AKA and $sSMS$ satisfy confidentiality of SMSs in isolation, *i.e.* whether the following processes do not reveal sms:

$$(sk_{SN}; C_{WS}[AKA^{SN}[0] \mid AKA^{MS}[0]]; \Phi_0) \quad (\alpha)$$

$$(sk_{SN}; C'_{WS}[sSMS^{SN} \mid sSMS^{MS}]; \emptyset) \qquad (\beta)$$

where $C'_{WS}[\_] \stackrel{\mathsf{def}}{=} C_{WS}[\mathtt{new}\ ck.\mathtt{let}\ xck_{SN} = ck\ \mathtt{in}\ \mathtt{let}\ xck_{MS} = ck\ \mathtt{in}\ \_]$.

Note that the composition context C_{WS} is the same as $\mathsf{fst}(C_U)$ (up to some renaming), thus Hypotheses (1-4) of Theorem 2 also hold and we derive the weak secrecy property by simply analysing this property on AKA and $sSMS$ in isolation.

We are left with verifying that AKA and $sSMS$ preserve weak secrecy of exchanged SMSs. AKA trivially does, since sms is not used in AKA. Using ProVerif we can show that $sSMS$ also preserves weak secrecy of SMSs.

Strong secrecy requires that an outside oberver does not distinguish the situation where sms_1 is sent, from the situation where sms_2 is sent, although he might know the content of sms_1 and sms_2. To model this requirement we consider the following composition context.

$$C_{SS}[\_] \stackrel{\mathsf{def}}{=} !\mathtt{new}\ IMSI.\ \mathtt{new}\ k_{IMSI}.\ !\mathtt{new}\ sqn.\ \mathtt{let}\ sms = \mathsf{diff}[sms_1, sms_2]\ \mathtt{in}\ \_$$

where sms_1 and sms_2 are two free names known to the attacker. This composition context differs on the content of the SMS being sent on the left-hand process and on the right-hand process. To check if the considered 3G protocols satisfy confidentiality of sent SMSs *w.r.t.* some initial intruder knowledge, *e.g.* $\Phi_0 = \{w_1 \rhd \mathsf{pk}(sk_{SN})\}$, one needs to check if the following biprocess satisfies diff-equivalence

$$(sk_{SN}; C_{SS}[AKA^{SN}[sSMS^{SN}] \mid AKA^{MS}[sSMS^{MS}]]; \Phi_0).$$

However, according to Theorem 4 we can instead check whether AKA and $sSMS$ satisfy strong secrecy of SMSs in isolation:

$$(sk_{SN}; C_{SS}[AKA^{+}]; \Phi_0)\ (\alpha) \quad \text{and} \quad (sk_{SN}; C'_{SS}[sSMS]; \emptyset)\ (\beta)$$

where AKA^{+} and $sSMS$ defined as for unlinkability, and

$$C'_{SS}[\_] \overset{\mathsf{def}}{=} C_{SS}[\texttt{new}\ ck.\texttt{let}\ xck_{SN} = ck\ \texttt{in}\ \texttt{let}\ xck_{MS} = ck\ \texttt{in}\ \_].$$

Indeed, AKA/Φ_0 and $sSMS/\Psi$ (for $\Psi = \emptyset$) are composable under $\mathcal{E}_0 = \{sk_{SN}\}$ and C_{SS}. Regarding the conditions of Theorem 4: *(i)* it is easy to see that AKA satisfies the abstractability property: both the MS and the SN compute the key ck and store it respectively in the assignment variables xck_{MS} and xck_{SN} by applying the function $\mathsf{f3}$, which is a one way function, to K_{IMSI} and $RAND$; *(ii)* using ProVerif we can show that the considered two protocols do not reveal xck_{MS} and xck_{SN}, and that AKA is actually a good key-exchange protocol. We are left with verifying that AKA and $sSMS$ preserve strong secrecy of exchanged SMSs. AKA trivially does, since the left and the right-hand processes are syntactically equal. Using ProVerif we can show that $sSMS$ also preserves strong secrecy of SMSs.

B Case study: e-passport

As mentioned in the introduction, many applications like electronic passports or mobile phones rely on several protocols running in composition (parallel, sequential, or nested). In this section, we show that our results can help in the analysis of this sort of complex system considering the e-passport application.

B.1 Protocols description

The information stored in the chip of the passport is organised in data groups (dg_1 to dg_{19}). For example, dg_5 contains a JPEG copy of the displayed picture, and dg_7 contains the displayed signature. The verification key $\mathsf{vk}(sk_P)$ of the passport, together with its certificate $\mathsf{sign}(\mathsf{vk}(sk_P), sk_{DS})$ issued by the Document Signer authority are stored in dg_{15}. The corresponding signing key sk_P is stored in a tamper resistant memory, and cannot be read or copied. For authentication purposes, a hash of all the dgs together with a signature on this hash value issued by the Document Signer authority are stored in a separate file, the Security Object Document:

$$sod \overset{\mathsf{def}}{=} \langle \mathsf{sign}(\mathsf{h}(dg_1, \ldots, dg_{19}), sk_{DS}),\ \mathsf{h}(dg_1, \ldots, dg_{19}) \rangle.$$

The ICAO standard specifies several protocols through which these information can be accessed [1].

The Basic Access Control ($\boldsymbol{BAC}$) protocol (see Figure 3) establishes a key seed *xkseed* from which two sessions keys *xkenc* and *xkmac* are derived. The purpose of *ksenc* and *ksmac* is to prevent skimming and eavesdropping on the subsequent communication with the e-passport (see below). The security of the BAC protocol relies on two master keys, *ke* and *km*, which are optically retrieved from the passport by the reader before executing the BAC protocol.

The reader initiates the protocol by sending a challenge to the passport and the passport replies with a random 64-bit string n_P. The reader then creates its own random nonce and some new random key material, both 64-bits. These are encrypted, along with the tag's nonce and sent back to the reader. A MAC is computed using the *km* key and sent along with the message, to ensure the message is received correctly. The tag receives this message, verifies the MAC, decrypts the message and checks that its nonce is correct; this guarantees to the tag that the message from the reader is not a replay of an old message. The tag then generates its own random 64-bits of key material and sends this back to the reader in a similar message, except this time the order of the nonces is reversed, this stops the readers message being replayed directly back to the reader. The reader checks the MAC and its nonce, and both the tag and the reader use the xor of the key material as the seed for a session key, with which to encrypt the rest of the session.

Once the BAC protocol has been successfully executed, the reader gains access to the information stored in the RFID tag through the Passive Authentication (*PA*) and the Active Authentication (AA) protocols that can be executed in any order.

The *PA* protocol (see Figure 4) is an authentication mechanism that proves that the content of the RFID chip is authentic. Through *PA* the reader retrieves the information stored in the *dg*s and the *sod*. It then verifies that the hash value stored in the *sod* corresponds to the one signed by the Document Signer authority. It further checks that this hash value is consistent with the received *dg*s.

The $\boldsymbol{AA}$ protocol (see Figure 5) is an authentication mechanism that prevents cloning of the passport chip. It relies on the fact that the secret key sk_P of the passport cannot be read or copied. The reader sends a random challenge to the passport, that has to return a signature on this challenge using its private signature key sk_P. The reader can then verify using the verification key $\mathsf{vk}(sk_P)$ that the signature was built using the expected passport key.

B.2 Privacy analysis

All three protocols BAC, *PA* and AA, rely on symmetric encryption and message authentication codes. Note that the only publicly known verification key is $\mathsf{vk}(sk_{DS})$ and is only used by the *PA* protocol. Thus, we can use our composition results, and in particular tour Theorems 3 and 4, to reason in a modular way

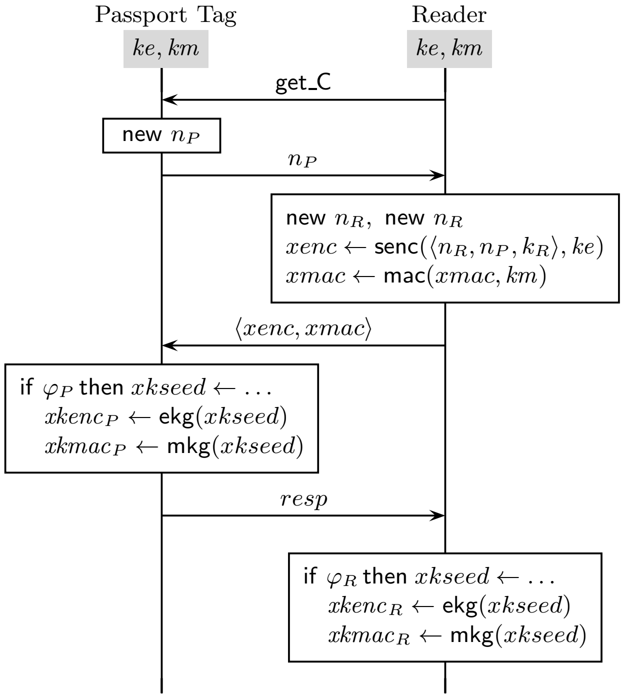


Fig. 3. The BAC protocol

about the privacy guarantees provided by the tagged version of the e-passport application.

In [4], as sequential composition could not be handled, the analysis of the e-passports application had to exclude the execution of the BAC protocol. Instead, it was assumed that the keys $kenc$ and $kmac$ were "magically" pre-shared. With our sequential composition result (Theorem 4), we avoid this unsafe abstraction, as we can now consider the execution of the BAC protocol for the establishment of these two keys. In this way, we are here able to complete the analysis of the e-passport application.

According to the ICAO standard, the reader optically retrieves the passport's master keys ke and km before executing the BAC protocol to establish the key seed for $kenc$ and $kmac$. The reader can then decide to execute PA and/or AA in any order. Formally, this corresponds to the sequential composition of the BAC protocol and of the PA and AA protocols composed in parallel. This system can be modelled in our calculus as follows:

$$P \stackrel{\mathsf{def}}{=} \mathtt{new}\ sk_{DS}.\ !\mathtt{new}\ ke.\ \mathtt{new}\ km.\ \mathtt{new}\ sk_P.\ \mathtt{new}\ id.\ \mathtt{new}\ sig.\ \mathtt{new}\ pic.\ \ldots$$
$$!(BAC^R[PA^R \mid AA^R] \mid BAC^P[PA^P \mid AA^P])$$

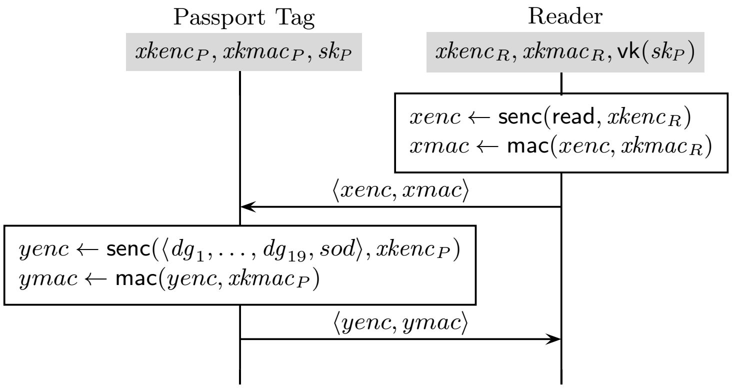


Fig. 4. Passive Authentication protocol

where id, sig, pic, ... represent the name, the signature, the displayed picture, *etc* of the e-passport's owner, *i.e.* the data stored in the dgs (1-14) and (16-19). The subprocesses BAC^P, PA^P and AA^P (*resp.* BAC^R, PA^R and AA^R) model one session of the passport's (*resp.* reader's) side of the BAC, PA and AA protocols respectively. The name sk_{DS} models the signing key of the Document Signing authority used in all passports. Each passport (identified by its master keys ke and km, its signing key sk_P, the owner's name, picture, signature, ...) can run multiple times the BAC protocol followed by the PA and AA protocols in any order.

To express strong anonymity, we need on one hand to consider a system in which the particular e-passport with publicly known id_1, sig_1, pic_1, *etc.* is being executed, while on the other hand it is a different e-passport with publicly known id_2, sig_2, pic_2, *etc.* which is being executed. For this we consider the following composition context:

$$C_A[\_] \stackrel{\mathsf{def}}{=} \texttt{let}\ id = \mathsf{diff}[id_1, id_2]\ \texttt{in}\ \ \ldots !\_$$

This composition context differs in the e-passport being executed on the left-hand process and on the right-hand process. In other words, the systems satisfies anonymity if an observer cannot distinguish whether the e-passport with publicly known id_1, sig_1, pic_1, *etc.* is being executed, or another e-passport is being executed (with publicly known id_2, sig_2, pic_2, *etc.*)

To check if the tagged version of the e-passport application (we assume here that BAC, PA, and AA are colored using three distinct colors, and thus will be tagged in different ways) preserves strong anonymity, one thus needs to check if the following biprocess satisfies diff-equivalence:

$$(sk_{DS}; C_A[[\![BAC^R[PA^R \mid AA^R] \mid BAC^P[PA^P \mid AA^P]]\!]]; \Phi_0)$$

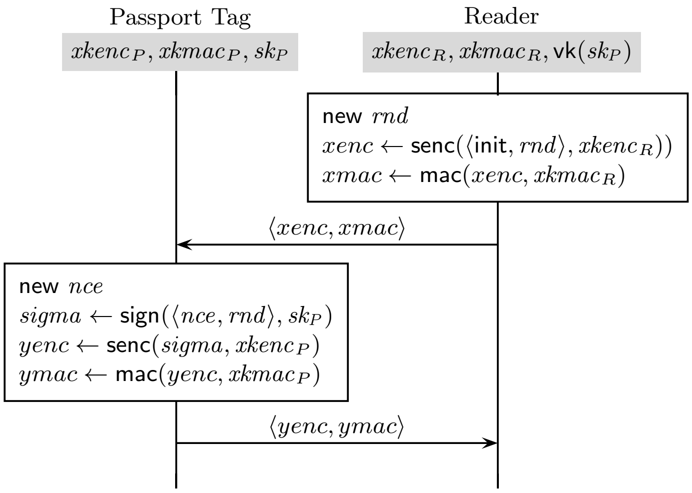


Fig. 5. Active Authentication protocol

We can instead check whether BAC, PA and AA satisfy anonymity in isolation, *i.e.* if the following three diff-equivalences hold:

$$(sk_{DS}; \texttt{new}\ d.\ C_A[[\![BAC^{+}]\!]]; \emptyset)\ (\alpha)$$
$$(sk_{DS}; C'_A[[\![PA^R \mid PA^P]\!]]; \Phi_0)\ \ (\beta)$$
$$(sk_{DS}; C'_A[[\![AA^R \mid AA^P]\!]]; \emptyset)\ \ (\gamma)$$

where $BAC^{+} \stackrel{\mathsf{def}}{=} BAC^R[\texttt{out}(d, (xkenc_R, xkmac_R))] \mid BAC^P[\texttt{out}(d, (xkenc_P, xkmac_P))]$
$\quad\quad \texttt{in}(d, x).\ \texttt{in}(d, y).\ \texttt{if}\ x = y\ \texttt{then}\ 0\ \texttt{else}\ 0$

$C'_A[\_] \stackrel{\mathsf{def}}{=} C_A[C''_A[\_]]$

$C''_A[\_] \stackrel{\mathsf{def}}{=} \mathsf{new}\ kenc.\ \mathsf{new}\ kmac$
$\quad \texttt{let}\ (xkenc_R, xkmac_R) = (kenc, kmac)\ \texttt{in}$
$\quad \texttt{let}\ (xkenc_P, xkmac_P) = (kenc, kmac)\ \texttt{in}\ \_$

Then, applying Theorem 3 to (β) and (γ) we derive that the following biprocess satisfies diff-equivalence:

$$(sk_{DS}; C'_A[[\![PA^R \mid AA^R \mid PA^P \mid AA^P]\!]]; \Phi_0)\ (\delta)$$

and applying Theorem 4 to (α) and (δ), we derive the required diff-equivalence:

$$(sk_{DS}; C_A[[\![BAC^R[PA^R \mid AA^R] \mid BAC^P[PA^P \mid AA^P]]\!]]; \Phi_0)$$

Indeed, let $P = BAC^R[0] | BAC^P[0]$; and $Q = C''_A[PA \mid AA]$, and assume that Ψ is the empty frame. We can check that P/Ψ and Q/Φ_0 are composable under $\mathcal{E}_0 = \{sk_{DS}\}$ and C_A (according to Definition 6). Note that $fn(P) \cap fn(Q) \cap bn(C_A) = \emptyset$, and thus the last condition trivially holds. Furthermore, using ProVerif we can show that properties (α) and γ are also satisfied. Unfortunately, ProVerif does not terminate when given the script corresponding to equivalence

(β). Note that ProVerif does not terminate when given the script corresponding to the hole system either. At this point our only solution would be to rely on a manual proof. Our composition results have allowed us to reduce a big equivalence that existing tools cannot handle, to a much smaller one.

C Sharing primitives via tagging

We recall in this section the tagging scheme as presented in [4]. However, since we would like to be able to iterate our composition results (in order to compose *e.g.* three protocols), we consider a fixed set of colors (not only two), and we allow a process to be colored with many colors. Actually, a colored process is a process with a color assigned to each of its action. This gives us enough flexibility to allow different kinds of compositions, and to iterate our composition results.

We consider a family of signatures $\Sigma_1, \dots, \Sigma_p$ disjoint from each other and disjoint from Σ_0. In order to tag a process, we introduce a new family of signatures $\Sigma_1^{\mathsf{tag}}, \dots, \Sigma_p^{\mathsf{tag}}$. For each $i \in \{1, \dots, p\}$, we have that $\Sigma_i^{\mathsf{tag}} = \{\mathsf{tag}_i, \mathsf{untag}_i\}$ where tag_i and untag_i are two function symbols of arity 1 that we will use for tagging. The role of the tag_i function is to tag its argument with the tag i. The role of the untag_i function is to remove the tag. To model this interaction between tag_i and untag_i, we consider the equational theory: $\mathsf{E}_{\mathsf{tag}_i} = \{\mathsf{untag}_i(\mathsf{tag}_i(x)) = x\}$.

For our composition result, we will assume that the two protocols we want to compose only share symbols in Σ_0. Thus, for this, we split the set $\{1, \dots, p\}$ into two disjoint sets α and β. Given a subset $\gamma \subseteq \{1, \dots, p\}$, we denote:

$$\begin{array}{lll} \Sigma_\gamma \overset{\mathsf{def}}{=} \bigcup_{i\in\gamma} \Sigma_i & \Sigma_\gamma^{\mathsf{tag}} \overset{\mathsf{def}}{=} \bigcup_{i\in\gamma} \Sigma_i^{\mathsf{tag}} & \Sigma_\gamma^{+} \overset{\mathsf{def}}{=} \Sigma_\gamma \cup \Sigma_\gamma^{\mathsf{tag}} \\ \mathsf{E}_\gamma \overset{\mathsf{def}}{=} \bigcup_{i\in\gamma} \mathsf{E}_i & \mathsf{E}_\gamma^{\mathsf{tag}} \overset{\mathsf{def}}{=} \bigcup_{i\in\gamma} \mathsf{E}_i^{\mathsf{tag}} & \mathsf{E}_\gamma^{+} \overset{\mathsf{def}}{=} \mathsf{E}_\gamma \cup \mathsf{E}_\gamma^{\mathsf{tag}} \end{array}$$

Definition 10. *Let $i \in \{1, \dots, p\}$, and u be a term built over $\Sigma_i \cup \Sigma_0$. The i-tagged version of u, denoted $[u]_i$ is defined as follows:*

$$\begin{array}{ll} [\mathsf{senc}(u,v)]_i \overset{\mathsf{def}}{=} \mathsf{senc}(\mathsf{tag}_i([u]_i), [v]_i) & [\mathsf{sdec}(u,v)]_i \overset{\mathsf{def}}{=} \mathsf{untag}_i(\mathsf{sdec}([u]_i, [v]_i)) \\ [\mathsf{aenc}(u,v)]_i \overset{\mathsf{def}}{=} \mathsf{aenc}(\mathsf{tag}_i([u]_i), [v]_i) & [\mathsf{adec}(u,v)]_i \overset{\mathsf{def}}{=} \mathsf{untag}_i(\mathsf{adec}([u]_i, [v]_i)) \\ [\mathsf{sign}(u,v)]_i \overset{\mathsf{def}}{=} \mathsf{sign}(\mathsf{tag}_i([u]_i), [v]_i) & [\mathsf{check}(u,v)]_i \overset{\mathsf{def}}{=} \mathsf{untag}_i(\mathsf{check}([u]_i, [v]_i)) \\ [\mathsf{h}(u)]_i \overset{\mathsf{def}}{=} \mathsf{h}(\mathsf{tag}_i([u]_i)) & [\mathsf{f}(u_1, \dots, u_n)]_i \overset{\mathsf{def}}{=} \mathsf{f}([u_1]_i, \dots, [u_n]_i) \textit{ otherwise.} \end{array}$$

Note that we do not tag the pairing function symbol (this is actually useless), and we do not tag the pk and vk function symbols. Note that tagging pk and vk would lead us to consider an unrealistic modelling for asymmetric keys. This definition is extended as expected to formulas φ (those involved in conditionals) by applying the transformation on each term that occurs in φ.

Example 10. Let $\Sigma_1 = \{\mathsf{f}, \mathsf{g}\}$, and consider the terms $u = \mathsf{senc}(\mathsf{g}(r), k)$ and $v = \mathsf{f}(\mathsf{sdec}(y, k), r)$ built on $\Sigma_1 \cup \Sigma_0$. We have that $[u]_1 = \mathsf{senc}(\mathsf{tag}_1(\mathsf{g}(r)), k)$, and $[v]_1 = \mathsf{f}(\mathsf{untag}_1(\mathsf{sdec}(y, k)), r)$.

We also introduce the following notion that allows us to associate a color to a term that is not necessarily well-tagged.

Definition 11. *Let u be a term. We define* $\mathsf{tagroot}(u)$*, namely the tag of the root of u as follows:*

- $\mathsf{tagroot}(u) = \bot$ *when* $u \in \mathcal{N} \cup \mathcal{X}$*;*
- $\mathsf{tagroot}(u) = i$ *if* $u = \mathsf{f}(u_1, \dots, u_n)$ *and either* $\mathsf{f} \in \Sigma_i \cup \Sigma_i^{\mathsf{tag}}$*, or* $\mathsf{f} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$ *and* $u_1 = \mathsf{tag}_i(u'_1)$ *for some* u'_1*.*
- $\mathsf{tagroot}(u) = 0$ *otherwise.*

Before extending the notion of tagging to processes, we have to express the tests that are performed by an agent when he receives a message that is supposed to be tagged. This is the purpose of $\mathsf{test}_i(u)$ that represents the tests which ensure that every projection and every untagging performed by an agent during the computation of u is successful.

Definition 12. *Let $i \in \{1, \dots, p\}$, and u be a term built on $\Sigma_i^+ \cup \Sigma_0$. We define* $\mathsf{test}_i(u)$ *as follows:*

- $\mathsf{test}_i(u) \stackrel{\mathsf{def}}{=} \mathsf{test}_i(u_1) \wedge \mathsf{test}_i(u_2) \wedge \mathsf{tag}_i(\mathsf{untag}_i(u)) = u$ *when* $u = \mathsf{g}(u_1, u_2)$ *with* $\mathsf{g} \in \{\mathsf{sdec}, \mathsf{adec}, \mathsf{check}\}$
- $\mathsf{test}_i(u) \stackrel{\mathsf{def}}{=} \mathsf{test}_i(u_1) \wedge u_1 = \langle \mathsf{proj}_1(u_1), \mathsf{proj}_2(u_1) \rangle$ *when* $u = \mathsf{proj}_j(u_1)$ *with* $j \in \{1, 2\}$
- $\mathsf{test}_i(u) \stackrel{\mathsf{def}}{=} \mathsf{true}$ *when u is a name or a variable*
- $\mathsf{test}_i(u) \stackrel{\mathsf{def}}{=} \mathsf{test}_i(u_1) \wedge \dots \wedge \mathsf{test}_i(u_n)$ *otherwise (with $u = f(u_1, \dots, u_n)$).*

This definition is extended as expected to formulas φ, *i.e.* $\mathsf{test}_i(\varphi) \stackrel{\mathsf{def}}{=} \bigwedge_{u=v \in \varphi} \mathsf{test}_i(u) \wedge \mathsf{test}_i(v)$.

Example 11. Again, consider $u = \mathsf{senc}(\mathsf{g}(r), k)$ and $v = \mathsf{f}(\mathsf{sdec}(y, k), r)$. We have that:

- $\mathsf{test}_1([u]_1) = \mathsf{true}$
- $\mathsf{test}_1([v]_1) = \mathsf{tag}_1(\mathsf{untag}_1(\mathsf{sdec}(y, k))) = \mathsf{sdec}(y, k)$

We consider colored plain processes meaning that initially the actions of a plain process will be annotated with a color, *i.e.* an integer in $\{1, \dots, p\}$. The actions that need to be annotated are those that involve some composed terms, *i.e.* inputs, outputs, conditionals, and assignments. An action colored by $i \in \{1, \dots, p\}$ can only contain function symbol from Σ_i. Given a set $\gamma \subseteq \{1, \dots, p\}$, we say than an action is colored with γ if this action is colored by $i \in \{1, \dots, p\}$. For colored plain processes, the transformation $[\![P]\!]$ is defined as follows:

$$[\![0]\!] \stackrel{\mathsf{def}}{=} 0 \qquad [\![!P]\!] \stackrel{\mathsf{def}}{=} ![\![P]\!] \qquad [\![\mathtt{new}\ k.P]\!] \stackrel{\mathsf{def}}{=} \mathtt{new}\ k.[\![P]\!] \qquad [\![P \mid Q]\!] \stackrel{\mathsf{def}}{=} [\![P]\!] \mid [\![Q]\!]$$

$$[\![\mathtt{in}(u,x)^i.P]\!] \stackrel{\mathsf{def}}{=} \mathtt{in}(u,x)^i.[\![P]\!] \quad [\![[x := v]^i.P]\!] \stackrel{\mathsf{def}}{=} (\mathtt{if}\ \mathsf{test}_i([v]_i)\ \mathtt{then}\ [x := [v]_i]^i.[\![P]\!])^i$$

$$[\![\mathtt{out}(u,v)^i.Q]\!] \stackrel{\mathsf{def}}{=} (\mathtt{if}\ \mathsf{test}_i([v]_i)\ \mathtt{then}\ \mathtt{out}(u,[v]_i)^i.[\![Q]\!])^i$$

$$[\![(\mathtt{if}\ \varphi\ \mathtt{then}\ P\ \mathtt{else}\ Q)^i]\!] \stackrel{\mathsf{def}}{=} (\mathtt{if}\ \varphi_{\mathsf{test}}\ \mathtt{then}\ (\mathtt{if}\ [\varphi]_i\ \mathtt{then}\ [\![P]\!]\ \mathtt{else}\ [\![Q]\!]))^i\ \mathtt{else}\ 0)^i$$
$$\text{where } \varphi_{\mathsf{test}} = \mathsf{test}_i([\varphi]_i)$$

Roughly, instead of simply outputting a term v, a process will first perform some tests to check that the term is correctly tagged and he will output its i-tagged version $[v]_i$. For an assignment, we will also check that the term is correctly tagged. For a conditional, the process will first check that the terms involved in the test φ are correctly tagged before checking that the test is satisfied. The annotations that occur on a plain process do not affect its semantics.

Definition 13. *Consider a set $\gamma \subseteq \{1, \dots, p\}$. Consider a plain process P built over $\Sigma_\gamma^+ \cup \Sigma_0$. We say that P is tagged if there exists a colored plain process Q built over Σ_γ such that $P = [\![Q]\!]$.*

D Biprocesses

The semantics of biprocesses is defined via a relation $\xrightarrow{\ell}_{\mathsf{bi}}$ that expresses when and how a biprocess may evolve. Intuitively, a biprocess reduces if and only if both sides of the biprocess reduce in the same way: a communication succeeds on both sides, a conditional has to be evaluated in the same way in both sides too. When the two sides of the biprocess reduce in different ways, the biprocess blocks. The semantics of biprocesses is formally described in Figure 6.

E The disjoint case for a trace

Composition usually works well in the so-called disjoint case, *i.e.* when the protocols under study do not share any secrets. The goal of this section is to show that we can map any trace corresponding to an execution of a protocol (with some sharing) to another trace which corresponds to an execution of a "disjoint case" (where protocols do not share any secrets) preserving static equivalence. We need a strong mapping to ensure that processes evolve simultaneously, and we rely for this on the notion of *biprocesses*.

We will see in this section that the composition of processes sharing some secrets (the so-called shared case) behaves as if they did not share any secret (the so-called disjoint case), provided that the shared secrets are never revealed and processes are tagged.

$$(\mathcal{E}; \{\texttt{if } \mathsf{diff}(\varphi_L, \varphi_R) \texttt{ then } Q_1 \texttt{ else } Q_2\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau}_{\mathsf{bi}} (\mathcal{E}; Q_1 \uplus \mathcal{P}; \Phi; \sigma) \quad \text{(THEN)}$$
$$\text{if } u\sigma =_{\mathsf{E}} v\sigma \text{ for each } u = v \in \varphi_L \cup \varphi_R$$

$$(\mathcal{E}; \{\texttt{if } \mathsf{diff}(\varphi_L, \varphi_R) \texttt{ then } Q_1 \texttt{ else } Q_2\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau}_{\mathsf{bi}} (\mathcal{E}; Q_2 \uplus \mathcal{P}; \Phi; \sigma) \quad \text{(ELSE)}$$
$$\text{if } u_L\sigma \neq_{\mathsf{E}} v_L\sigma \text{ for some } u_L = v_L \in \varphi_L$$
$$\text{and } u_R\sigma \neq_{\mathsf{E}} v_R\sigma \text{ for some } u_R = v_R \in \varphi_R$$

$$(\mathcal{E}; \{\mathtt{out}(c, u).Q_1; \mathtt{in}(c, x).Q_2\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau}_{\mathsf{bi}} (\mathcal{E}; Q_1 \uplus Q_2 \uplus \mathcal{P}; \Phi; \sigma \cup \{x \mapsto u\sigma\}) \quad \text{(COMM)}$$

$$(\mathcal{E}; \{[x := v].Q\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau}_{\mathsf{bi}} (\mathcal{E}; Q \uplus \mathcal{P}; \Phi; \sigma \cup \{x \mapsto v\sigma\}) \quad \text{(ASSGN)}$$

$$(\mathcal{E}; \{\mathtt{in}(c, z).Q\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\mathtt{in}(c,M)}_{\mathsf{bi}} (\mathcal{E}; Q \uplus \mathcal{P}; \Phi; \sigma \cup \{z \mapsto u\}) \quad \text{(IN)}$$
$$\text{if } c \notin \mathcal{E},\ M\Phi = u,\ fv(M) \subseteq \mathrm{dom}(\Phi) \text{ and } fn(M) \cap \mathcal{E} = \emptyset$$

$$(\mathcal{E}; \{\mathtt{out}(c, u).Q\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\nu w_n.\mathsf{out}(c, w_n)}_{\mathsf{bi}} (\mathcal{E}; Q \uplus \mathcal{P}; \Phi \cup \{w_n \rhd u\sigma\}; \sigma) \quad \text{(OUT-T)}$$
$$\text{if } c \notin \mathcal{E},\ u \text{ is a term of base type, and } w_n \text{ is a variable such that } n = |\Phi| + 1$$

$$(\mathcal{E}; \{\mathtt{new}\ n.Q\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau}_{\mathsf{bi}} (\mathcal{E} \cup \{n\}; Q \uplus \mathcal{P}; \Phi; \sigma) \quad \text{(NEW)}$$

$$(\mathcal{E}; \{!Q\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau}_{\mathsf{bi}} (\mathcal{E}; \{!Q; Q\rho\} \uplus \mathcal{P}; \Phi; \sigma) \quad \text{(REPL)}$$
$$\text{where } \rho \text{ is used to rename variables in } bv(Q)$$
$$\text{(resp. names in } bn(Q)) \text{ with fresh variables (resp. names).}$$

$$(\mathcal{E}; \{P_1 \mid P_2\} \uplus \mathcal{P}; \Phi; \sigma) \xrightarrow{\tau}_{\mathsf{bi}} (\mathcal{E}; \{P_1, P_2\} \uplus \mathcal{P}; \Phi; \sigma) \quad \text{(PAR)}$$

where n is a name, c is a name of channel type (here we can have $c = \mathsf{diff}(c_1, c_2)$), u, v are terms that may contain the diff operator, and x, z are variables. The term M used in the IN rule is a term that does not contain any occurrence of the diff operator. The attacker has to do the same computation in both sides.

Fig. 6. Semantics for biprocesses

E.1 Material for combination

To handle the different signatures and equational theories, we consider the notion of *ordered rewriting*. It has been shown that by applying the *unfailing completion procedure* to E where $\mathsf{E} = \mathsf{E}_1 \uplus \mathsf{E}_2 \uplus \ldots \mathsf{E}_p$ is the union of disjoint equational theories (Σ_i, E_i) (for all i, j, we have that $\Sigma_i \cap \Sigma_j = \emptyset$), we can derive a (possibly infinite) set of equations $\mathcal{O}$ such that on ground terms:

1. the relations $=_{\mathcal{O}}$ and $=_{\mathsf{E}}$ are equal,
2. the rewriting system $\rightarrow_{\mathcal{O}}$ is convergent.

Since the relation $\rightarrow_{\mathcal{O}}$ is convergent on ground terms, we define $M{\downarrow_{\mathsf{E}}}$ (or briefly $M{\downarrow}$) as the unique normal form of the ground term M for $\rightarrow_{\mathcal{O}}$. These notations are extended as expected to sets of terms.

We now introduce our notion of *factors* and state some properties on them w.r.t. the different equational theories. A similar notion is also used in [?].

Definition 14 (factors). *Let $M \in \mathcal{T}(\Sigma, \mathcal{N} \cup \mathcal{X})$. The* factors *of M, denoted $Fct(M)$, are the maximal syntactic subterms of M that are alien to M*

Lemma 2. *Let M be a ground term such that all its factors are in normal form and* $\mathsf{root}(M) \in \Sigma_i$. *Then*

- *either* $M{\downarrow} \in Fct(M) \cup \{n_{min}\}$,
- *or* $\mathsf{root}(M{\downarrow}) \in \Sigma_i$ *and* $Fct(M{\downarrow}) \subseteq Fct(M) \cup \{n_{min}\}$.

Lemma 3. *Let t be a ground term with $t = C_1[u_1, \ldots, u_n]$ where C_1 is a context built on Σ_i, $i \in \{1, \ldots, p\}$ and the terms $u_1, \ldots, u_n$ are the factors of t in normal form. Let C_2 be a context built on Σ_i (possibly a hole) such that $t{\downarrow} = C_2[u_{j_1}, \ldots, u_{j_k}]$ with $j_1, \ldots, j_k \in \{0 \ldots n\}$ and $u_0 = n_{min}$ (the existence is given by Lemma 2). We have that for all ground terms $v_1, \ldots, v_n$ in normal form and alien to t, if*

$$\text{for every } q, q' \in \{1 \ldots n\} \text{ we have } u_q = u_{q'} \Leftrightarrow v_q = v_{q'}$$

then $C_1[v_1, \ldots, v_n]{\downarrow} = C_2[v_{j_1}, \ldots, v_{j_k}]$ *with* $v_0 = n_{min}$.

A proof of these lemmas can be found in [**?**,**?**].

E.2 Generic composition result

We consider two sets α, β such that $\alpha \cup \beta = \{1, \ldots, p\}$ and $\alpha \cap \beta = \emptyset$. We consider a plain colored process P built on $\Sigma_\alpha \cup \Sigma_\beta \cup \Sigma_0$ without replication and such that $bn(P) = fv(P) = \emptyset$. This means that P is a process with no free variables, and we assume that it contains no name restrictions (*i.e.* no `new` instructions).

Example 12. We consider the process P_{DH} as given in Example 2 but we replace

- the 0 at the end of P_A with $Q_A = \mathtt{new}\, s_A.\mathtt{out}(c, \mathsf{senc}_{DH}(s_A, x_A))$, and
- the 0 at the end of P_B with $Q_B = \mathtt{new}\, s_B.\mathtt{out}(c, \mathsf{senc}_{DH}(s_B, x_B))$.

Intuitively, once the Diffie-Hellman key has been established and stored in x_A (resp. x_B), each participant will use it to encrypt a fresh secret, namely s_A or s_B, and then send it to the other participant.

Note that when function symbols of Σ_0 are used by only one of the protocols to compose, we can either consider them as part of Σ_0 and so they will be tagged, or they can be put into distinct signatures (using renaming as above) and so they will not be tagged. The composition theorem can be applied both ways.

To avoid confusion between the encryption schemes that processes can share, *i.e.* the function symbols in Σ_0, and the asymmetric encryption used in P_{DH} but not used in Q_A and Q_B, we will rename them by $\mathsf{aenc}_{DH}, \mathsf{adec}_{DH}, \mathsf{pk}_{DH}$. Thus, we consider $p = 2$, $\alpha = \{1\}$, $\beta = \{2\}$ with $(\Sigma_\alpha, \mathsf{E}_\alpha) = (\Sigma_{\mathsf{DH}}, \mathsf{E}_{\mathsf{DH}})$ with

- $\Sigma_{\mathsf{DH}} = \{\mathsf{aenc}_{DH}, \mathsf{adec}_{DH}, \mathsf{pk}_{DH}, \mathsf{f}, \mathsf{g}\}$, and
- $\mathsf{E}_{\mathsf{DH}} = \{\mathsf{adec}_{DH}(\mathsf{aenc}_{DH}(x, \mathsf{pk}_{DH}(y)), y) = x,\ \ \mathsf{f}(\mathsf{g}(x), y) = \mathsf{f}(x, \mathsf{g}(y))\}$

whereas $\Sigma_\beta = \{\mathsf{senc}_{DH}, \mathsf{sdec}_{DH}\}$ and $\mathsf{E}_\beta = \{\mathsf{sdec}_{DH}(\mathsf{senc}_{DH}(x, y), y) = x\}$. This equational theory is used to model symmetric encryption/decryption, *i.e.* the primitives used in the processes Q_A and Q_B.

Now, we consider $P = P'_A \mid P'_B$ where:

– $P'_A = \mathtt{out}(c, \mathsf{aenc}_{DH}(\langle n_A, \mathsf{g}(r_A)\rangle, \mathsf{pk}_{DH}(sk_B))).\mathtt{in}(c, y_A).$
$\quad\mathtt{if}\ \mathsf{proj}_1(\mathsf{adec}(y_A, sk_A)) = n_A$
$\quad\mathtt{then}\ [x_A := \mathsf{f}(\mathsf{proj}_2(\mathsf{adec}_{DH}(y_A, sk_A)), r_A)].\mathtt{out}(c, \mathsf{senc}_{DH}(s_A, x_A))$

– $P'_B = \mathtt{in}(c, y_B).\mathtt{out}(c, \mathsf{aenc}_{DH}(\langle \mathsf{proj}_1(\mathsf{adec}_{DH}(y_B, sk_B)), \mathsf{g}(r_B)\rangle, \mathsf{pk}_{DH}(sk_A))).$
$\quad[x_B := \mathsf{f}(\mathsf{proj}_2(\mathsf{adec}_{DH}(y_B, sk_B)), r_B)].\mathtt{out}(c, \mathsf{senc}_{DH}(s_B, x_B))$

Note that $bn(P) = fv(P) = \emptyset$. We choose to color the three first actions of P'_A (resp. P'_B) with $1 \in \alpha$, and the remaining ones (*i.e.* those that come from Q_A and Q_B) with $2 \in \beta$.

We denote $fn^\gamma(P)$ the set of free names of P that occur in actions colored with γ, and $fv^\gamma(P)$ the set of variables of P that occur in an action colored with γ, and that are not bound by an action colored with γ. We consider a set $\mathcal{E}_0$ of names such that $fn^\alpha(P) \cap fn^\beta(P) \cap \mathcal{E}_0 = \emptyset$. This means that each name in $\mathcal{E}_0$ can only occur in one type of actions (those colored α or those colored β). We denote $z_1^\alpha, \ldots, z_k^\alpha$ (resp. $z_1^\beta, \ldots, z_l^\beta$) the variables occurring in the left-hand side of an assignment colored α (resp. β), *i.e.* the variable x such that the action $[x := v]$ occurs in P and is colored α (resp. β). We assume that $fv^\alpha(P) \subseteq \{z_1^\beta, \ldots, z_l^\beta\}$ and $fv^\beta(P) \subseteq \{z_1^\alpha, \ldots, z_k^\alpha\}$.

These conditions ensure that sharing between the parts of the process which are colored in different ways is only possible via the assignment variables. This is not a real limitation but this allows us to easily keep track of the shared data.

Example 13. Continuing our example, we have $fn^\alpha(P) = \{r_A, r_B, n_A, sk_A, sk_B\}$ and $fn^\beta(P) = \{s_A, s_B\}$. Regarding variables: $fv^\alpha(P) = \emptyset$, whereas $fv^\beta(P) = \{x_A, x_B\}$.

Let $\mathcal{E}_0 = fn^\alpha(P) \cup fn^\beta(P)$. To follow the same notation as those introduced in this section, we may want to rename x_A with z_1^α and x_B with z_2^α. Note that $fv^\beta(P) \subseteq \{z_1^\alpha, z_2^\alpha\}$.

Let $\mathcal{E}_\alpha = \{n_1^\alpha, \ldots, n_k^\alpha\}$ and $\mathcal{E}_\beta = \{n_1^\beta, \ldots, n_l^\beta\}$ be two sets of fresh names of base type such that $\mathcal{E}_\alpha \cap \mathcal{E}_\beta = \emptyset$. We define ρ_α and ρ_β as follows:

– $\mathrm{dom}(\rho_\alpha) = \{z_1^\beta, \ldots, z_l^\beta\}$, $\mathrm{dom}(\rho_\beta) = \{z_1^\alpha, \ldots, z_k^\alpha\}$;
– $\rho_\alpha(z_i^\beta) = n_i^\beta$ for each $i \in \{1, \ldots, l\}$; and
– $\rho_\beta(z_i^\alpha) = n_i^\alpha$ for each $i \in \{1, \ldots, k\}$.

We do not assume that names in $\mathcal{E}_\alpha$ (resp. $\mathcal{E}_\beta$) are distinct. For instance, we may have $n_j^\alpha = n_{j'}^\alpha$ for some $j \neq j'$.

Given a colored plain process P, we denote by $\delta_{\rho_\alpha,\rho_\beta}(P)$, the process obtained by applying ρ_α on actions colored α, and ρ_β on actions colored β. This transformation maps the shared case to a particular disjoint case.

Example 14. Let $\mathcal{E}_\alpha = \{k^\alpha\}$ and $\mathcal{E}_\beta = \emptyset$, and consider the function ρ_β defined as follows: $\rho_\beta(z_1^\alpha) = \rho_\beta(z_2^\alpha) = k^\alpha$. Applying $\delta_{\rho_\alpha,\rho_\beta}$ on P gives us $D_A \mid D_B$ where:

– $D_A = \mathtt{out}(c, \mathsf{aenc}_{DH}(\langle n_A, \mathsf{g}(r_A)\rangle, \mathsf{pk}_{DH}(sk_B))).\mathtt{in}(c, y_A).$
$\mathtt{if}\ \mathsf{proj}_1(\mathsf{adec}_{DH}(y_A, sk_A)) = n_A$
$\mathtt{then}\ [x_A := \mathsf{f}(\mathsf{proj}_2(\mathsf{adec}_{DH}(y_A, sk_A)), r_A)].\mathtt{out}(c, \mathsf{senc}_{DH}(s_A, k^\alpha))$
– $D_B = \mathtt{in}(c, y_B).\mathtt{out}(c, \mathsf{aenc}_{DH}(\langle \mathsf{proj}_1(\mathsf{adec}_{DH}(y_B, sk_B)), \mathsf{g}(r_B)\rangle, \mathsf{pk}_{DH}(sk_A))).$
$[x_B := \mathsf{f}(\mathsf{proj}_2(\mathsf{adec}_{DH}(y_B, sk_B)), r_B)].\mathtt{out}(c, \mathsf{senc}_{DH}(s_B, k^\alpha))$

Note that there is no sharing anymore between the part of the process colored α and the part of the process colored β.

Actually, the disjoint case obtained using the transformation $\delta_{\rho_\alpha,\rho_\beta}$ behaves as the shared case but only along executions that are *compatible* with the chosen abstractions, *i.e.* executions that preserve the equalities and the inequalities among assignment variables as done by the chosen abstraction. This notion is formally defined as follows:

Let A be any extended process derived from $(\mathcal{E}_\alpha \uplus \mathcal{E}_\beta \uplus \mathcal{E}_0; [\![P]\!]; \emptyset)$, *i.e.* such that $(\mathcal{E}_\alpha \uplus \mathcal{E}_\beta \uplus \mathcal{E}_0; [\![P]\!]; \emptyset) \overset{\mathsf{tr}}{\Longrightarrow} A$. For $\gamma \in \{\alpha, \beta\}$, we say that ρ_γ is *compatible* with $A = (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ when:

1. for all $x, y \in \mathrm{dom}(\sigma) \cap \mathrm{dom}(\rho_\gamma)$, we have that $x\sigma =_{\mathsf{E}} y\sigma$ if, and only if, $x\rho_\gamma = y\rho_\gamma$; and
2. for all $z \in \mathrm{dom}(\rho_\gamma)$, either $\mathsf{tagroot}(z\sigma\!\downarrow) = \perp$ or $\mathsf{tagroot}(z\sigma\!\downarrow) \notin \gamma \cup \{0\}$.

We say that $(\rho_\alpha, \rho_\beta)$ is *compatible* with A when both ρ_α and ρ_β are compatible with A. For $\gamma \in \{\alpha, \beta\}$, we define the *extension* of ρ_γ, denoted ρ_γ^+, as follows:

– $\mathrm{dom}(\rho_\gamma^+) = \mathrm{dom}(\rho_\gamma) \cup \{x\sigma\!\downarrow \mid x \in \mathrm{dom}(\rho_\gamma)\}$, and
– for any $x \in \mathrm{dom}(\rho_\gamma)$, $\rho_\gamma^+(x) \overset{\mathsf{def}}{=} \rho(x)$ and $\rho_\gamma^+(x\sigma) \overset{\mathsf{def}}{=} \rho_\gamma(x)$.

Before stating our generic composition result, we have also to formalize the fact that the shared keys are not revealed. Since sharing is performed via the assignment variables, we say that A_0 *does not reveal the value of its assignments* w.r.t. $(\rho_\alpha, \rho_\beta)$ if for any extended process $A = (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ derived from A_0 and such that $(\rho_\alpha, \rho_\beta)$ is compatible with A, we have:

$$\mathtt{new}\,\mathcal{E}.\Phi \not\vdash k \text{ for any } k \in K_\alpha \cup K_\beta$$

where for all $\gamma \in \{\alpha, \beta\}$, $K_\gamma = \{t, \mathsf{pk}(t), \mathsf{vk}(t) \mid z \in \mathrm{dom}(\sigma) \cap \mathrm{dom}(\rho_\gamma)$ and $(t = z\sigma$ or $t = z\rho_\gamma)\}$.

Theorem 5. *Let P be a plain colored process as described above, and B_0 be an extended colored biprocess such that:*

– $S_0 = (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta \uplus \mathcal{E}_0; [\![P]\!]; \emptyset; \emptyset) \overset{\mathsf{def}}{=} \mathsf{fst}(B_0)$,
– $D_0 = (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta \uplus \mathcal{E}_0; P_D; \emptyset; \emptyset) \overset{\mathsf{def}}{=} \mathsf{snd}(B_0)$, *and*
– $P_D = \delta_{\rho_\alpha,\rho_\beta}([\![P]\!])$ *for some* $(\rho_\alpha, \rho_\beta)$ *compatible with* D_0, *and*
– D_0 *does not reveal its assignments w.r.t.* $(\rho_\alpha, \rho_\beta)$.

We have that:

1. *For any extended process $S = (\mathcal{E}_S; \mathcal{P}_S; \Phi_S; \sigma_S)$ such that $S_0 \overset{\mathsf{tr}}{\Longrightarrow} S$ with $(\rho_\alpha, \rho_\beta)$ compatible with S, there exists a biprocess B and an extended process $D = (\mathcal{E}_D; \mathcal{P}_D; \Phi_D; \sigma_D)$ such that $B_0 \overset{\mathsf{tr}}{\Longrightarrow}_{\mathsf{bi}} B$, $\mathsf{fst}(B) = S$, $\mathsf{snd}(B) = D$, and $\mathtt{new}\,\mathcal{E}_S.\Phi_S \sim \mathtt{new}\,\mathcal{E}_D.\Phi_D$.*
2. *For any extended process $D = (\mathcal{E}_D; \mathcal{P}_D; \Phi_D; \sigma_D)$ such that $D_0 \overset{\mathsf{tr}}{\Longrightarrow} D$ with $(\rho_\alpha, \rho_\beta)$ compatible with D, there exists a biprocess B and an extended process $S = (\mathcal{E}_S; \mathcal{P}_S; \Phi_S; \sigma_S)$ such that $B_0 \overset{\mathsf{tr}}{\Longrightarrow}_{\mathsf{bi}} B$, $\mathsf{fst}(B) = S$, $\mathsf{snd}(B) = D$, and $\mathtt{new}\,\mathcal{E}_S.\Phi_S \sim \mathtt{new}\,\mathcal{E}_D.\Phi_D$.*

This theorem is proved by induction on the length of the derivation. For this, a strong correspondence between the process S_0 (shared case) and D_0 (disjoint case) has to be maintained along the derivation, and the transformation $\delta_{\rho_\alpha,\rho_\beta}$ has to be extended to allow replacements also in σ and Φ. The rest of this section is dedicated to the proof of this theorem.

Example 15. Going back to our running example, and forming a biprocess with $S_0 = (\mathcal{E}_0 \cup \{k^\alpha\}; P'_A \mid P'_B; \emptyset; \emptyset)$ and $D_0 = (\mathcal{E}_0 \cup \{k^\alpha\}; D_A \mid D_B; \emptyset; \emptyset)$, Theorem 5 gives us that these two processes behave in the same way when considering executions that are compatible with the chosen abstraction ρ_β, *i.e.* executions that instantiate x_A and x_B by the same value.

A similar result as the one stated in Theorem 5 was proved in [12]. Here, we consider in addition else branches, and we consider a richer common signature. Moreover, relying on the notion of biprocess, we show a strong link between the shared case and the disjoint case, and we prove in addition static equivalence of the resulting frames.

E.3 Name replacement

Now that we have fixed some notations, we have to explain how the replacement will be applied on the shared process to extract the disjoint case. Actually a same term will be abstracted differently depending on the context which is just above it.

Definition 15. *Let $(\rho^+_\alpha, \rho^+_\beta)$ be two functions from terms of base type to names of base type. Let $\delta_\gamma^{\rho^+_\alpha,\rho^+_\beta}$, or shortly δ_γ, $(\gamma \in \{\alpha, \beta\})$ be the functions on terms that is defined as follows:*

$$\delta_\gamma(u) = u{\downarrow}\rho^+_\gamma \ \text{ when } \begin{cases} u{\downarrow} \in \mathrm{dom}(\rho^+_\gamma) \\ \text{and } \mathsf{tagroot}(u) \notin \gamma \cup \{0\} \end{cases}$$

Otherwise, we have that $\delta_\gamma(u) = u$ when u is a name or a variable; and $\delta_\gamma(\mathsf{f}(t_1, \ldots, t_k))$ is equal to

- $\mathsf{f}(\delta_\gamma(t_1), \ldots, \delta_\gamma(t_k))$ *if* $\mathsf{tagroot}(\mathsf{f}(t_1, \ldots, t_n)) = 0$;
- $\mathsf{f}(\delta_\alpha(t_1), \ldots, \delta_\alpha(t_k))$ *if* $\mathsf{tagroot}(\mathsf{f}(t_1, \ldots, t_n)) \in \alpha$.
- $\mathsf{f}(\delta_\beta(t_1), \ldots, \delta_\beta(t_k))$ *if* $\mathsf{tagroot}(\mathsf{f}(t_1, \ldots, t_n)) \in \beta$.

Definition 16 (Factor for Σ_0). *Let u be a term. We define $Fct_{\Sigma_0}(u)$ the factors of a term u for Σ_0 as the maximal syntactic subterms v of u such that* $\mathsf{tagroot}(v) \neq 0$.

Let σ be a substitution. We consider a pair $(\rho_\alpha, \rho_\beta)$ as defined in Section E.2 and compatible with σ. We denote $(\rho_\alpha^+, \rho_\beta^+)$ the extension of $(\rho_\alpha, \rho_\beta)$ w.r.t. σ. Thanks to compatibility, ρ_α^+ (resp. ρ_β^+) is injective on $\mathrm{dom}(\rho_\alpha^+) \smallsetminus \{z_1^\beta, \ldots, z_l^\beta\}$ (resp. $\mathrm{dom}(\rho_\beta^+) \smallsetminus \{z_1^\alpha, \ldots, z_k^\alpha\}$). Moreover, we also have that for all $u \in \mathrm{dom}(\rho_\alpha^+) \smallsetminus \{z_1^\beta, \ldots, z_l^\beta\}$ (resp. $\mathrm{dom}(\rho_\beta^+) \smallsetminus \{z_1^\alpha, \ldots, z_k^\alpha\}$), either $\mathsf{tagroot}(u) = \bot$ or $\mathsf{tagroot}(u) \notin \alpha \cup \{0\}$ (resp. $\mathsf{tagroot}(u) \notin \beta \cup \{0\}$).

Lemma 4. *Let t_1 and t_2 be ground terms in normal form such that $(fn(t_1) \cup fn(t_2)) \cap (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta) = \emptyset$. We have that:*

$$t_1 = t_2 \textit{ if, and only if, } \delta_\gamma(t_1) = \delta_\gamma(t_2)$$

where $\gamma \in \{\alpha, \beta\}$.

Proof. The right implication is trivial. We consider the left implication, and we prove the result by induction on $\max(|t_1|, |t_2|)$ when $\gamma = \alpha$. The other case $\gamma = \beta$ can be handled in a similar way.

Base case $\max(|t_1|, |t_2|) = 1$*:* In such a case, we have that $t_1, t_2 \in \mathcal{N}$. We first assume that $\delta_\alpha(t_1)$ (and thus also $\delta_\alpha(t_2)$) is in $\mathcal{E}_\alpha \uplus \mathcal{E}_\beta$. By hypothesis, we know that t_2 and t_1 do not use names in $\mathcal{E}_\alpha \uplus \mathcal{E}_\beta$. Therefore, by definition of δ_α, we can deduce that $t_1, t_2 \in \mathrm{dom}(\rho_\alpha^+)$ and $t_1\rho_\alpha^+ = t_2\rho_\alpha^+$, and thus $t_1 = t_2$ thanks to ρ_α^+ being injective on $\mathrm{dom}(\rho_\alpha^+) \smallsetminus \{z_1^\beta, \ldots, z_l^\beta\}$. Now, we assume that $\delta_\alpha(t_1)$ (and thus also $\delta_\alpha(t_2)$) is not in $\mathcal{E}_\alpha \uplus \mathcal{E}_\beta$. In such a case, by definition of δ_α, we have that $\delta_\alpha(t_1) = t_1$ and $\delta_\alpha(t_2) = t_2$, and thus $t_1 = t_2$.

Inductive step $\max(|t_1|, |t_2|) > 1$*:* Assume w.l.o.g. that $|t_1| > 1$. Thus, there exists a symbol function f and terms $u_1, \ldots u_n$ such that $t_1 = \mathsf{f}(u_1, \ldots u_n)$. We do a case analysis on t_1 which is in normal form.

<u>Case $t_1 \in \mathrm{dom}(\rho_\alpha^+)$:</u> In such a case, $\delta_\alpha(t_1) = \delta_\alpha(t_2) = n$ for some $n \in \mathcal{E}_\alpha$. By hypothesis, we know that t_2 and t_1 do not use names in $\mathcal{E}_\alpha$, and we have that $t_1\rho_\alpha^+ = t_2\rho_\alpha^+$. Therefore, we necessarily have that $t_1 = t_2$.

<u>Case $t_1 \notin \mathrm{dom}(\rho_\alpha^+)$:</u> We do a new case analysis on t_1.

Case $\mathsf{f} \in \Sigma_i^+$ *for some* $i \in \{1, \ldots, p\}$*:* Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. In such a case, we have that $\delta_\alpha(t_1) = \mathsf{f}(\delta_\gamma(u_1), \ldots, \delta_\gamma(u_n))$. But $\delta_\alpha(t_2) = \delta_\alpha(t_1)$ and by definition of δ_α, it implies that there exist $v_1, \ldots, v_n$ such that $t_2 = \mathsf{f}(v_1, \ldots, v_n)$ and $\mathsf{f}(\delta_\gamma(v_1), \ldots, \delta_\gamma(v_n)) = \delta_\alpha(t_2)$. Thus we have that $\delta_\gamma(v_j) = \delta_\gamma(u_j)$ for all $j \in \{1, \ldots, n\}$. Furthermore, since t_1 and t_2 are in normal form and not using names in $\mathcal{E}_\alpha \uplus \mathcal{E}_\beta$, we also know that u_j and v_j are in normal form and not using names in $\mathcal{E}_\alpha \uplus \mathcal{E}_\beta$, for every j. Since, we have that $\max(|t_1|, |t_2|) > \max(|u_j|, |v_j|)$, for any j, by our inductive hypothesis, we can deduce that $u_j = v_j$, for all j and so $t_1 = \mathsf{f}(u_1, \ldots, u_n) = \mathsf{f}(v_1, \ldots, v_n) = t_2$.

Case $t_1 = \mathsf{f}(\mathsf{tag}_i(w_1), w_2)$ *with* $i \in \{1, \ldots, p\}$ *and* $\mathsf{f} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\}$*:* Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. In such a case, we know that $\delta_\alpha(t_1) = \mathsf{f}(\mathsf{tag}_i(\delta_\gamma(w_1)), \delta_\gamma(w_2))$. But we know that $\delta_\alpha(t_2) = \delta_\alpha(t_1) = \mathsf{f}(\mathsf{tag}_i(\delta_\gamma(w_1)), \delta_\gamma(w_2))$. Thus thanks to t_2 being in normal form and by definition of δ_α, it implies that there exists v_1 and v_2 such that $t_2 = \mathsf{f}(\mathsf{tag}_i(v_1), v_2)$ and so $\delta_\alpha(t_2) = \mathsf{f}(\mathsf{tag}_i(\delta_\gamma(v_1)), \delta_\gamma(v_2))$. Thus, we have that $\delta_\gamma(v_1) = \delta_\gamma(u_1)$ and $\delta_\gamma(v_2) = \delta_\gamma(u_2)$. Moreover, t_1 and t_2 being in normal form and not using names in $\mathcal{E}_\alpha \uplus \mathcal{E}_\beta$, so are u_j and v_j for $j \in \{1, 2\}$, so we can apply inductive hypothesis and conclude that $v_1 = u_1$ and $v_2 = u_2$ and so $t_1 = t_2$.

Case $t_1 = \mathsf{h}(\mathsf{tag}_i(w_1))$ *with* $i \in \{1, \ldots, p\}$*:* This case is analogous to the previous one.

Case $\mathsf{f} \in \Sigma_0$ *and* $\mathsf{root}(u_1) \neq \mathsf{tag}_i$*,* $i = 1 \ldots p$*:* By definition of δ_α, we can deduce that $\delta_\alpha(t_1) = \mathsf{f}(\delta_\alpha(u_1), \ldots, \delta_\alpha(u_n))$. Since $\delta_\alpha(t_1) = \delta_\alpha(t_2)$, we can deduce that the top symbol of t_2 is also f and so there exists $v_1, \ldots, v_n$ such that $t_2 = \mathsf{f}(v_1, \ldots, v_n)$. In the previous cases, we showed that if $\mathsf{f} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$ and the top symbol of v_1 is tag_j for some $j \in \{1, \ldots, p\}$ then $\delta_\alpha(t_1) = \delta_\alpha(t_2)$ implies that the top symbol of u_1 is also tag_j. Thus, thanks to our hypothesis, we can deduce that either $\mathsf{f} \notin \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$ or the top symbol of v_1 is different from tag_j for some $j \in \{1, \ldots, p\}$. Hence by definition of δ_α, we can deduce that $\delta_\alpha(t_2) = \mathsf{f}(\delta_\alpha(v_1), \ldots, \delta_\alpha(v_n))$ and so $\delta_\alpha(v_j) = \delta_\alpha(u_j)$ for all $j \in \{1, \ldots, n\}$. Moreover, t_1 and t_2 being in normal form and not using names in $\mathcal{E}_\alpha \uplus \mathcal{E}_\beta$, implies that so are u_j and v_j for all $j \in \{1, \ldots, n\}$. We can thus apply our inductive hypothesis and conclude that $u_j = v_j$ for all $j \in \{1, \ldots, n\}$ and so $t_1 = t_2$.

Lemma 5. *Let t_1 and t_2 be ground terms in normal form such that $(fn(t_1) \cup fn(t_2)) \cap (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta) = \emptyset$. We have that:*

$$\delta_\alpha(t_1) = \delta_\beta(t_2) \textit{ implies that } t_1 = t_2.$$

Proof. We prove the result by induction on $|\delta_\alpha(t_1)|$.

Base case $|\delta_\alpha(t_1)| = 1$*:* Since $\delta_\alpha(t_1) = \delta_\beta(t_2)$, $\mathcal{E}_\alpha \cap \mathcal{E}_\beta = \emptyset$, and t_1, t_2 do not use names in $\mathcal{E}_\alpha \uplus \mathcal{E}_\beta$, we necessarily have that $t_1 \notin \mathrm{dom}(\rho_\alpha^+)$ and $t_2 \notin \mathrm{dom}(\rho_\beta^+)$. Hence, we have that $\delta_\alpha(t_1) = t_1$ and $\delta_\beta(t_2) = t_2$. This allows us to conclude.

Inductive step $|\delta_\alpha(t_1)| > 1$*:* In that case, we have that $\delta_\alpha(t_1) = \mathsf{f}(u_1, \ldots, u_n) = \delta_\beta(t_2)$. Assume that $\mathsf{f} \in \Sigma_i \cup \Sigma_{\mathsf{tag}_i}$ for some $i \in \{1, \ldots, p\}$. Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. By definition of δ_α and δ_β, we can deduce that $\mathsf{root}(t_1) = \mathsf{f} = \mathsf{root}(t_2)$. Furthermore, if we assume that $t_1 = \mathsf{f}(v_1, \ldots, v_n)$ and $t_2 = \mathsf{f}(w_1, \ldots, w_n)$, we would have $\delta_\gamma(v_j) = \delta_\gamma(w_j)$ for all $j \in \{1, \ldots, n\}$. By Lemma 4, we deduce that $v_j = w_j$ for all $j \in \{1, \ldots, n\}$. Hence, we conclude that $t_1 = t_2$. Assume now that $\mathsf{f} \in \Sigma_0$. According to the definition of δ_α and δ_β, there exists $v_1, \ldots, v_n$ and $w_1, \ldots, w_n$ such that $t_1 = \mathsf{f}(v_1, \ldots, v_n)$, $t_2 = \mathsf{f}(w_1, \ldots, w_n)$ and $\delta_{\gamma_1}(v_j) = \delta_{\gamma_2}(w_j)$, for some $\gamma_1, \gamma_2 \in \{\alpha, \beta\}$. Moreover, t_1 and t_2 being in normal form and not using names in $\mathcal{E}_\alpha \uplus \mathcal{E}_\beta$ implies that so are v_j and w_j for all $j \in \{1, \ldots, n\}$. Now, either $\gamma_1 = \gamma_2$ and so by Lemma 4, we have that $v_j = w_j$, else $\gamma_1 \neq \gamma_2$ but then by our inductive hypothesis, we also have $v_j = w_j$. Hence we conclude that $t_1 = t_2$.

Lemma 6. *Let u be a ground term in normal form such that $fn(u) \cap (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta) = \emptyset$. Let $\gamma \in \{\alpha, \beta\}$. We have that:*

- $\delta_\gamma(u)$ *is in normal form; and*
- *either* $\mathsf{root}(\delta_\gamma(u)) = \mathsf{root}(u)$ *or* $\mathsf{root}(\delta_\gamma(u)) = \bot$.
- *either* $\mathsf{tagroot}(\delta_\gamma(u)) = \mathsf{tagroot}(u)$ *or* $\mathsf{tagroot}(\delta_\gamma(u)) = \bot$.

Proof. We prove this result by induction on $|u|$ and we assume w.l.o.g. that $\gamma = \alpha$.

Base case $|u| = 1$*:* In such a case, we have that $u \in \mathcal{N}$, and we also have that $\delta_\alpha(u) \in \mathcal{N}$ and so $\delta_\alpha(u)$ is in normal form with the same root as u, namely $\bot$. Moreover, we have $\mathsf{tagroot}(\delta_\gamma(u)) = \bot$.

Inductive $|u| > 1$*:* Assume first that $u{\downarrow} \in \mathrm{dom}(\rho^+_\alpha)$ and $\mathsf{tagroot}(u) \notin \alpha \cup \{0\}$. Hence by definition of δ_α, we have that $\delta_\alpha(u) \in \mathcal{E}_\alpha$. Thus, we trivially obtain that $\delta_\alpha(u)$ is in normal form, $\mathsf{root}(\delta_\alpha(u)) = \bot$ and $\mathsf{tagroot}(\delta_\alpha(u)) = \bot$.

Otherwise, we distinguish two cases:

<u>*Case 1.*</u> We have that $u = C[u_1, \ldots, u_n]$ where C is built on $\Sigma_j \cup \Sigma_{\mathsf{tag}_j}$ with $j \in \{1, \ldots, p\}$, C is different from a hole, u_k are factors in normal form of u, $k = 1 \ldots n$. Let $\varepsilon \in \{\alpha, \beta\}$ such that $j \in \varepsilon$. Hence, since $u \notin \mathrm{dom}(\rho^+_\alpha)$, then by definition of δ_α, we deduce that $\delta_\alpha(u) = C[\delta_\varepsilon(u_1), \ldots, \delta_\varepsilon(u_n)]$. Since C is not a hole, thanks to our inductive hypothesis on $u_1, \ldots, u_n$, we have that $\delta_\varepsilon(u_1), \ldots, \delta_\varepsilon(u_n)$ are in normal form and $\delta_\varepsilon(u_1), \ldots, \delta_\varepsilon(u_n)$ are factors of $\delta_\alpha(u)$. Thus, since u is in normal form, we have that $C[u_1, \ldots, u_n]{\downarrow} = C[u_1, \ldots, u_n]$. By Lemmas 4 and 3, we deduce that

$$C[\delta_\varepsilon(u_1), \ldots, \delta_\varepsilon(u_n)]{\downarrow} = C[\delta_\varepsilon(u_1), \ldots, \delta_\varepsilon(u_n)]$$

i.e. $\delta_\alpha(u){\downarrow} = \delta_\alpha(u)$.

Furthermore, we also have that $\mathsf{root}(\delta_\alpha(u)) = \mathsf{root}(u)$ and $\mathsf{tagroot}(\delta_\alpha(u)) = \mathsf{tagroot}(u)$.

<u>*Case 2.*</u> We have that $u = \mathsf{f}(v_1, \ldots, v_n)$ for some $\mathsf{f} \in \Sigma_0$. By definition of δ_α there exists $\varepsilon \in \{\alpha, \beta\}$ such that $\delta_\alpha(u) = \mathsf{f}(\delta_\varepsilon(v_1), \ldots, \delta_\varepsilon(v_m))$. We do a case analysis on f:

Case $\mathsf{f} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{pk}, \mathsf{sign}, \mathsf{vk}, \mathsf{h}, \langle\ \rangle\}$*:* In this case, we have that $\delta_\alpha(u){\downarrow} = \mathsf{f}(\delta_\varepsilon(v_1){\downarrow}, \ldots, \delta_\varepsilon(v_m){\downarrow})$. Since by inductive hypothesis, $\delta_\varepsilon(v_k)$ is in normal form, for all $k \in \{1, \ldots, m\}$, we can deduce that $\delta_\alpha(u)$ is also in normal form and $\mathsf{root}(\delta_\alpha(u)) = \mathsf{f} = \mathsf{root}(u)$. If $\mathsf{f} \in \{\mathsf{pk}, \mathsf{vk}, \langle\ \rangle\}$ then we trivially have that $\mathsf{tagroot}(\delta_\alpha(u)) = \mathsf{tagroot}(u)$. Let's focus on $\mathsf{f} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\}$. If $\mathsf{tagroot}(u) \notin \{0\}$ then it means that $\mathsf{root}(v_1) = \mathsf{tag}_i$ for some $i \in \varepsilon$. But by definition of δ_ε, we would have that $\mathsf{root}(\delta_\varepsilon(v_1)) = \mathsf{tag}_i$. Hence $\mathsf{tagroot}(\delta_\alpha(u)) = \mathsf{tagroot}(u)$. Now if $\mathsf{tagroot}(u) \notin \{0\}$, it means that $\mathsf{root}(v_1)\ {/}\{\mathsf{tag}_1 \ldots \mathsf{tag}_p\}$. But by inductive hypothesis, $\mathsf{root}(\delta_\varepsilon(v_1)) = \bot$ or $\mathsf{root}(\delta_\varepsilon(v_1)) = \mathsf{root}(v_1)$ and so we can conclude that $\mathsf{tagroot}(\delta_\alpha(u)) \in \{0\}$.

Case $\mathsf{f} = \mathsf{sdec}$*:* Then $m = 2$, and by definition of δ_α, we have that $\delta_\alpha(u) = \mathsf{sdec}(\delta_\alpha(v_1), \delta_\alpha(v_2))$. Thus, in such a case, we have that $\mathsf{root}(\delta_\alpha(u)) = \mathsf{f} =$

$\mathsf{root}(u)$. By inductive hypothesis, we have that $\delta_\alpha(v_1)$ and $\delta_\alpha(v_2)$ are both in normal form. Assume that sdec cannot be reduced, *i.e.* $\delta_\alpha(u)\!\downarrow = \mathsf{sdec}(\delta_\alpha(v_1)\!\downarrow, \delta_\alpha(v_2)\!\downarrow) = \mathsf{sdec}(\delta_\alpha(v_1), \delta_\alpha(v_2))$. Thus the result holds. Otherwise, if sdec can be reduced, there exist w_1, w_2 with $\delta_\alpha(v_1) = \mathsf{senc}(w_1, w_2)$ and $\delta_\alpha(v_2) = w_2$. By definition of δ_α, there must exist $\varepsilon' \in \{\alpha, \beta\}$, and w'_1, w'_2 such that $\delta_\alpha(v_1) = \mathsf{senc}(\delta_{\varepsilon'}(w'_1), \delta_{\varepsilon'}(w'_2))$, $v_1 = \mathsf{senc}(w'_1, w'_2)$, $w_1 = \delta_{\varepsilon'}(w'_1)$ and $w_2 = \delta_{\varepsilon'}(w'_2)$. Thus, we have that $\delta_\alpha(v_2) = \delta\varepsilon'(w'_2)$. Thanks to Lemmas 4 and 5, we have that $v_2 = w'_2$. Hence, $u = \mathsf{sdec}(\mathsf{senc}(w'_1, w'_2), w'_2)$. But in such a case, we would have that u is not in normal form which contradicts our hypothesis.

At last, since $\mathsf{root}(\delta_\alpha(u)) = \bot$ or $\mathsf{root}(\delta_\alpha(u)) = \mathsf{root}(u) = \mathsf{sdec}$ then we can deduce that $\mathsf{tagroot}(\delta_\alpha(u)) = \bot$ or $\mathsf{tagroot}(\delta_\alpha(u)) = \mathsf{tagroot}(u) = 0$.

The cases where $\mathsf{f} = \mathsf{check}$ or $\mathsf{f} = \mathsf{adec}$ are analogous to the previous one.

E.4 δ_α and δ_β on tagged term

Let σ_0 be a ground substitution. Similarly to the previous section, we consider a pair $(\rho_\alpha, \rho_\beta)$ as defined in Section E.2 and compatible with σ_0. We denote $(\rho^+_\alpha, \rho^+_\beta)$ the extension of $(\rho_\alpha, \rho_\beta)$ w.r.t. this substitution. We also denote by $\mathcal{E}_\alpha$ and $\mathcal{E}_\beta$ the respective image of ρ_β and ρ_α, and we assume that σ_0 does not use any name in $\mathcal{E}_\alpha$ and $\mathcal{E}_\beta$.

Thanks to compatibility, ρ^+_α (resp. ρ^+_β) is injective on $\mathrm{dom}(\rho^+_\alpha) \smallsetminus \{z^\beta_1, \ldots, z^\beta_l\}$ (resp. $\mathrm{dom}(\rho^+_\beta) \smallsetminus \{z^\alpha_1, \ldots, z^\alpha_k\}$). Moreover, we also have that for all $z \in \{z^\beta_1, \ldots, z^\beta_l\}$ (resp. $z \in \{z^\alpha_1, \ldots, z^\alpha_k\}$), either $\mathsf{tagroot}(z\sigma_0\!\downarrow) = \bot$ or $\mathsf{tagroot}(z\sigma_0\!\downarrow) \notin \alpha \cup \{0\}$ (resp. $\mathsf{tagroot}(z\sigma_0\!\downarrow) \notin \beta \cup \{0\}$).

Let $i \in \{1, \ldots, p\}$. Let $u \in \mathcal{T}(\Sigma_i \cup \Sigma_{\mathsf{tag}_i} \cup \Sigma_0, \mathcal{N} \cup \mathcal{X})$. As defined in Section C, $\mathsf{test}_i(u)$ is a conjunction of elementary formulas (equalities between terms). Given a substitution σ such that $fv(u) \subseteq \mathrm{dom}(\sigma)$, we say that σ satisfies $t_1 = t_2$, denoted $\sigma \vDash t_1 = t_2$, if $t_1\sigma\!\downarrow = t_2\sigma\!\downarrow$.

At last, for all substitution σ, for all $\gamma \in \{\alpha, \beta\}$, we denote by $\delta_\gamma(\sigma)$ the substitution such that $\mathrm{dom}(\sigma) = \mathrm{dom}(\delta_\gamma(\sigma))$ and for all $x \in \mathrm{dom}(\delta_\gamma(\sigma))$, $x\delta_\gamma(\sigma) = \delta_\gamma(x\sigma)$.

Lemma 7. *Let $u \in \mathcal{T}(\Sigma_i \cup \Sigma_0, \mathcal{N} \cup \mathcal{X})$ for some $i \in \{1, \ldots, p\}$. Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$ and σ_0 be a ground substitution such that $fv(u) \subseteq \mathrm{dom}(\sigma_0)$. Moreover, assume that u does not use names in $\mathcal{E}_\alpha \uplus \mathcal{E}_\beta$. We have that:*

- $\delta_\gamma([u]_i(\sigma_0\!\downarrow)) = \delta_\gamma([u]_i)\delta_\gamma(\sigma_0\!\downarrow)$*; and*
- *If* $\sigma_0 \vDash \mathsf{test}_i([u]_i)$ *then* $\delta_\gamma([u]_i(\sigma_0\!\downarrow))\!\downarrow = \delta_\gamma([u]_i\sigma_0\!\downarrow)$.

Proof. Let σ be the substitution $\sigma_0\!\downarrow$. We prove the two results separately. First, we show by induction on $|u|$ that $\delta_\gamma([u]_i\sigma) = \delta_\gamma([u]_i)\delta_\gamma(\sigma)$:

Base case $|u| = 1$: In this case, $u \in \mathcal{N} \cup \mathcal{X}$. If $u \in \mathcal{N}$ then we have that $[u]_i = u$ and so $[u]_i\sigma = u$ and $\delta_\gamma(u) \in \mathcal{N}$. Thus, we have that $\delta_\gamma([u]_i\sigma) = \delta_\gamma(u) = \delta_\gamma(u)\delta_\gamma(\sigma) = \delta_\gamma([u]_i)\delta_\gamma(\sigma)$. Otherwise, we have that $u \in \mathcal{X}$ and $[u]_i = u$.

W.l.o.g., we assume that $\gamma = \alpha$. First, if $u \notin \{z_1^\beta, \ldots, z_l^\beta\}$, then we have that $\delta_\alpha(u) = u$. Thus, $\delta_\alpha(u)\delta_\alpha(\sigma) = u\delta_\alpha(\sigma)$. Since $u \in \mathcal{X}$ and $fv(u) \subseteq \mathrm{dom}(\sigma)$, we have that $u\delta_\alpha(\sigma) = \delta_\alpha(u\sigma)$, thus $\delta_\alpha([u]_i\sigma) = \delta_\alpha(u\sigma) = u\delta_\alpha(\sigma) = \delta_\alpha(u)\delta_\alpha(\sigma) = \delta_\alpha([u]_i)\delta_\alpha(\sigma)$. Now, it remains the case where $u = z_j^\beta$ for some $j \in \{1, \ldots, l\}$. In such a case, we have that:

- $\delta_\alpha([z_j^\beta]_i\sigma) = \delta_\alpha(z_j^\beta\sigma) = n_j^\beta$, and
- $\delta_\alpha([z_j^\beta]_i)\delta_\alpha(\sigma) = \delta_\alpha(z_j^\beta)\delta_\alpha(\sigma) = n_j^\beta\delta_\alpha(\sigma) = n_j^\beta$.

Inductive step $|u| > 1|$, i.e. $u = \mathsf{f}(u_1, \ldots, u_n)$. We do a case analysis on f.

Case $\mathsf{f} \in \Sigma_i$: In such a case, $[u]_i = \mathsf{f}([u_1]_i, \ldots, [u_n]_i)$. By definition of δ_γ, $\delta_\gamma([u]_i\sigma) = \mathsf{f}(\delta_\gamma([u_1]_i\sigma), \ldots, \delta_\gamma([u_n]_i\sigma))$ and $\delta_\gamma([u]_i) = \mathsf{f}(\delta_\gamma([u_1]_i), \ldots, \delta_\gamma([u_n]_i))$. By our inductive hypothesis, we can deduce that for all $k \in \{1, \ldots, n\}$, we have that $\delta_\gamma([u_k]_i\sigma) = \delta_\gamma([u_k]_i)\delta_\gamma(\sigma)$. Thus, we can deduce that $\delta_\gamma([u]_i\sigma) = \mathsf{f}(\delta_\gamma([u_1]_i), \ldots, \delta_\gamma([u_n]_i))\delta_\gamma(\sigma) = \delta_\gamma([u]_i)\delta_\gamma(\sigma)$.

Case $\mathsf{f} \in \{\mathsf{aenc}, \mathsf{sign}, \mathsf{senc}\}$: In this case $n = 2$, and by definition of $[u]_i$, we have that $[u]_i = \mathsf{f}(\mathsf{tag}_i([u_1]_i), [u_2]_i)$. Thus, $\delta_\gamma([u]_i) = \mathsf{f}(\mathsf{tag}_i(\delta_\gamma([u_1]_i)), \delta_\gamma([u_2]_i))$ and $\delta_\gamma([u]_i\sigma) = \mathsf{f}(\mathsf{tag}_i(\delta_\gamma([u_1]_i\sigma)), \delta_\gamma([u_2]_i\sigma))$. But by our inductive hypothesis, we have $\delta_\gamma([u_k]_i\sigma) = \delta_\gamma([u_k]_i)\delta_\gamma(\sigma)$ with $k \in \{1, 2\}$. We conclude that

$$\begin{aligned}\delta_\gamma([u]_i\sigma) &= \mathsf{f}(\mathsf{tag}_i(\delta_\gamma([u_1]_i)\delta_\gamma(\sigma)), \delta_\gamma([u_2]_i)\delta_\gamma(\sigma)) \\ &= \delta_\gamma([u]_i)\delta_\gamma(\sigma).\end{aligned}$$

Case $\mathsf{f} = \mathsf{h}$: This case is analogous to the previous one and can be handled in a similar way.

Case $\mathsf{f} \in \{\mathsf{sdec}, \mathsf{adec}, \mathsf{check}\}$: In this case $n = 2$, and by definition of $[u]_i$, we have that $[u]_i = \mathsf{untag}_i(\mathsf{f}([u_1]_i, [u_2]_i))$. Thus, $\delta_\gamma([u]_i) = \mathsf{untag}_i(\mathsf{f}(\delta_\gamma([u_1]_i), \delta_\gamma([u_2]_i)))$ and $\delta_\gamma([u]_i\sigma) = \mathsf{untag}_i(\mathsf{f}(\delta_\gamma([u_1]_i\sigma), \delta_\gamma([u_2]_i\sigma)))$. Relying on our inductive hypothesis, we deduce that

$$\delta_\gamma([u_k]_i\sigma) = \delta_\gamma([u_k]_i)\delta_\gamma(\sigma) \text{ with } k \in \{1, 2\}.$$

We conclude that

$$\begin{aligned}\delta_\gamma([u]_i\sigma) &= \mathsf{untag}_i(\mathsf{f}(\delta_\gamma([u_1]_i), \delta_\gamma([u_2]_i)))\delta_\gamma(\sigma) \\ &= \delta_\gamma([u]_i)\delta_\gamma(\sigma).\end{aligned}$$

Otherwise, by definition of $[u]_i$, we have that:

- $[u]_i = \mathsf{f}([u_1]_i, \ldots, [u_n]_i)$, and
- $\delta_\gamma([u]_i) = \mathsf{f}(\delta_\gamma([u_1]_i), \ldots, \delta_\gamma([u_n]_i))$.

Thus, this case is similar to the case $\mathsf{f} \in \Sigma_i$. Hence the result holds.

We now prove the second property, *i.e.* if $\sigma \vDash \mathsf{test}_i([u]_i)$, then $\delta_\gamma([u]_i\sigma)\!\downarrow = \delta_\gamma([u]_i\sigma\!\downarrow)$. We prove the result by induction on $|u|$:

Base case $|u| = 1$*:* In this case, $u \in \mathcal{N} \cup \mathcal{X}$. In both cases, we have that $[u]_i = u$ and $\mathsf{test}_i(u) = \mathsf{true}$. If $u \in \mathcal{N}$, we know that $\delta_\gamma(u) \in \mathcal{N}$ and so $\delta_\gamma(u)\downarrow = \delta_\gamma(u)$. We also have that $u\sigma\downarrow = u\sigma = u$. This allows us to conclude that

$$\delta_\gamma(u\sigma)\downarrow = \delta_\gamma(u)\downarrow = \delta_\gamma(u) = \delta_\gamma(u\sigma\downarrow).$$

Otherwise, we have $u \in \mathcal{X}$. Since σ is is normal form, we deduce that $u\sigma\downarrow = u\sigma$. By Lemma 6, we also know that $\delta_\gamma(u\sigma\downarrow)\downarrow = \delta_\gamma(u\sigma\downarrow)$. Thus, we conclude that

$$\delta_\gamma(u\sigma\downarrow) = \delta_\gamma(u\sigma\downarrow)\downarrow = \delta_\gamma(u\sigma)\downarrow.$$

Inductive step $|u| > 1$*, i.e.* $u = \mathsf{f}(u_1, \ldots, u_n)$. We do a case analysis on f.

Case $\mathsf{f} \in \Sigma_i$*:* We have that $[u]_i = \mathsf{f}([u_1]_i, \ldots, [u_n]_i)$. Hence, we have that $\delta_\gamma([u]_i\sigma) = \mathsf{f}(\delta_\gamma([u_1]_i\sigma), \ldots, \delta_\gamma([u_n]_i\sigma))$ and so $\delta_\gamma([u]_i\sigma)\downarrow = \mathsf{f}(\delta_\gamma([u_1]_i\sigma)\downarrow, \ldots, \delta_\gamma([u_n]_i\sigma)\downarrow)\downarrow$. We have that $\mathsf{test}_i([u]_i) = \bigwedge_{j=1}^{n} \mathsf{test}_i([u_j]_i)$ which means that $\sigma \vDash \mathsf{test}_i([u_j]_i)$ for each $j \in \{1, \ldots, n\}$. By applying our inductive hypothesis on $u_1, \ldots, u_n$, we deduce that

$$\begin{aligned}\delta_\gamma([u]_i\sigma)\downarrow &= \mathsf{f}(\delta_\gamma([u_1]_i\sigma\downarrow), \ldots, \delta_\gamma([u_n]_i\sigma\downarrow))\downarrow \\ &= \delta_\gamma(\mathsf{f}([u_1]_i\sigma\downarrow, \ldots, [u_n]_i\sigma\downarrow))\downarrow\end{aligned}$$

Let $t = \mathsf{f}([u_1]_i\sigma\downarrow, \ldots, [u_n]_i\sigma\downarrow)$. We can assume that there exists a context C built on Σ_i such that $t = C[t_1, \ldots, t_m]$ with $Fct(t) = \{t_1, \ldots, t_m\}$ and $t_1, \ldots, t_m$ are in normal form. Thus, by Lemma 2, there exists a context D (possibly a hole) such that $t\downarrow = D[t_{j_1}, \ldots, t_{j_k}]$ with $j_1, \ldots, j_k \in \{0, \ldots, m\}$ and $t_0 = n_{min}$. Since $t_1, \ldots, t_m$ are in normal form and thanks to Lemma 6, we know that for all $k \in \{0, \ldots, m\}$, $\delta_\gamma(t_k)$ is also in normal form and its root is not in Σ_i. Hence, we can apply Lemma 3 such that $C[\delta_\gamma(t_1), \ldots, \delta_\gamma(t_m)]\downarrow = D[\delta_\gamma(t_{j_1}), \ldots, \delta_\gamma(t_{j_k})]$. But since C and D are both built upon Σ_i, we have that:

- $C[\delta_\gamma(t_1), \ldots, \delta_\gamma(t_m)]\downarrow = \delta_\gamma(C[t_1, \ldots, t_m])\downarrow$, and
- $D[\delta_\gamma(t_{j_1}), \ldots, \delta_\gamma(t_{j_k})] = \delta_\gamma(D[t_{j_1}, \ldots, t_{j_k}])$.

Hence, we can deduce that $\delta_\gamma(t)\downarrow = \delta_\gamma(t\downarrow)$. But we already know that $t\downarrow = [u]_i\sigma\downarrow$ and $\delta_\gamma(t)\downarrow = \delta_\gamma([u]_i\sigma)\downarrow$. Thus, we can conclude that $\delta_\gamma([u]_i\sigma)\downarrow = \delta_\gamma([u]_i\sigma\downarrow)$.

Case $\mathsf{f} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\}$*:* In such a case, we have that:

- $[u]_i = \mathsf{f}(\mathsf{tag}_i([u_1]_i), [u_2]_i)$, and
- $\mathsf{test}_i([u]_i) = \mathsf{test}_i([u_1]_i) \wedge \mathsf{test}_i([u_2]_i)$.

Hence, we have that $[u]_i\sigma\downarrow = \mathsf{f}(\mathsf{tag}_i([u_1]_i\sigma\downarrow), [u_2]_i\sigma\downarrow)$, and also $\delta_\gamma([u]_i\sigma)\downarrow = \mathsf{f}(\mathsf{tag}_i(\delta_\gamma([u_1]_i\sigma)\downarrow), \delta_\gamma([u_2]_i\sigma)\downarrow)$. By our inductive hypothesis on u_1 and u_2, we have that:

$$\delta_\gamma([u_k]_i\sigma)\downarrow = \delta_\gamma([u_k]_i\sigma\downarrow) \text{ with } k \in \{1, 2\}.$$

Hence, we can deduce that

$$\begin{aligned}\delta_\gamma([u]_i\sigma)\downarrow &= \mathsf{f}(\mathsf{tag}_i(\delta_\gamma([u_1]_i\sigma\downarrow)), \delta_\gamma([u_2]_i\sigma\downarrow)) \\ &= \delta_\gamma(\mathsf{f}(\mathsf{tag}_i([u_1]_i\sigma\downarrow), [u_2]_i\sigma\downarrow)) \\ &= \delta_\gamma([u]_i\sigma\downarrow).\end{aligned}$$

Case $\mathsf{f} = \mathsf{h}$*:* This case is analogous to the previous one and can be handled in a similar way.

Case $\mathsf{f} \in \{\mathsf{pk}, \mathsf{vk}, \langle\ \rangle\}$*:* In such a case, we have that:

- $[u]_i = \mathsf{f}([u_1]_i, \ldots, [u_n]_i)$ with $n \in \{1, 2\}$, and
- $\mathsf{test}_i([u]_i) = \wedge_{j=1}^{n} \mathsf{test}_i([u_j]_i)$.

We have that $[u]_i\sigma\downarrow = \mathsf{f}([u_1]_i\sigma\downarrow, \ldots, [u_n]_i\sigma\downarrow)$. Thus, this case is similar to the senc case and can be handled similarly.

Case $\mathsf{f} \in \{\mathsf{sdec}, \mathsf{adec}, \mathsf{check}\}$*:* In such a case, we have that:

- $[u]_i = \mathsf{untag}_i(\mathsf{f}([u_1]_i, [u_2]_i))$, and
- $\mathsf{test}_i([u]_i)$ is the following formula:

$$\begin{aligned}&(\mathsf{tag}_i(\mathsf{untag}_i(\mathsf{f}([u_1]_i, [u_2]_i))) = \mathsf{f}([u_1]_i, [u_2]_i))\\ &\wedge\ \mathsf{test}_i([u_1]_i)\ \wedge\ \mathsf{test}_i([u_2]_i)\end{aligned}$$

By hypothesis, we have that $\sigma \vDash \mathsf{test}_i([u]_i)$, thus $\mathsf{tag}_i(\mathsf{untag}_i(\mathsf{f}([u_1]_i, [u_2]_i)))\sigma\downarrow = \mathsf{f}([u_1]_i, [u_2]_i)\sigma\downarrow$. Hence, we deduce that the root function symbol f can be reduced and the root of the plaintext is tag_i. More formally, there exist v_1, v_2 such that:

- $\mathsf{f} = \mathsf{sdec}$: $[u_1]_i\sigma\downarrow = \mathsf{senc}(\mathsf{tag}_i(v_1), v_2)$, $[u_2]_i\sigma\downarrow = v_2$ and $[u]_i\sigma\downarrow = v_1$. This implies that:
$$\delta_\gamma([u_1]_i\sigma\downarrow) = \mathsf{senc}(\mathsf{tag}_i(\delta_\gamma(v_1)), \delta_\gamma(v_2)).$$
Thus, we can deduce that:
$$\begin{aligned}\mathsf{untag}_i(\mathsf{sdec}(\delta_\gamma([u_1]_i\sigma\downarrow), \delta_\gamma([u_2]_i\sigma\downarrow)))\downarrow &= \delta_\gamma(v_1)\\ &= \delta_\gamma([u]_i\sigma\downarrow)\end{aligned}$$
- $\mathsf{f} = \mathsf{adec}$: $[u_1]_i\sigma\downarrow = \mathsf{aenc}(\mathsf{tag}_i(v_1), \mathsf{pk}(v_2))$, $[u_2]_i\sigma\downarrow = v_2$, and $[u]_i\sigma\downarrow = v_1$.
- $\mathsf{f} = \mathsf{check}$: $[u_1]_i\sigma\downarrow = \mathsf{sign}(\mathsf{tag}_i(v_1), v_2)$, $[u_2]_i\sigma\downarrow = \mathsf{vk}(v_2)$, and $[u]_i\sigma\downarrow = v_1$.

In each case, we have that:

$$\mathsf{untag}_i(\mathsf{f}(\delta_\gamma([u_1]_i\sigma\downarrow), \delta_\gamma([u_2]_i\sigma\downarrow)))\downarrow = \delta_\gamma([u]_i\sigma\downarrow).$$

By inductive hypothesis, we have $\delta_\gamma([u_k]_i\sigma\downarrow) = \delta_\gamma([u_k]_i\sigma)\downarrow$ with $k \in \{1, 2\}$. We also have that:

$$\delta_\gamma([u]_i\sigma)\downarrow = \mathsf{untag}_i(\mathsf{f}(\delta_\gamma([u_1]_i\sigma)\downarrow, \delta_\gamma([u_2]_i\sigma)\downarrow))\downarrow.$$

This allows us to conclude that

$$\begin{aligned}\delta_\gamma([u]_i\sigma)\downarrow &= \mathsf{untag}_i(\mathsf{f}(\delta_\gamma([u_1]_i\sigma\downarrow), \delta_\gamma([u_2]_i\sigma\downarrow)))\downarrow\\ &= \delta_\gamma([u]_i\sigma\downarrow).\end{aligned}$$

Case $\mathsf{f} = \mathsf{proj}_j$, $j = 1, 2$*:* In such a case, we have that $n = 1$, and $[u]_i = \mathsf{f}([u_1]_i)$. Since $\sigma \vDash \mathsf{test}_i([u]_i)$, we have that there exist v_1, v_2 such that $[u_1]_i\sigma\downarrow = \langle v_1, v_2\rangle$ and so $\delta_\gamma([u]_i\sigma\downarrow) = \delta_\gamma(v_j)$. But by inductive hypothesis, we have that $\delta_\gamma([u_1]_i\sigma)\downarrow = \delta_\gamma([u_1]_i\sigma\downarrow) = \langle\delta_\gamma(v_1), \delta_\gamma(v_2)\rangle$. Hence, $\delta_\gamma([u]_i\sigma)\downarrow = \mathsf{f}(\delta_\gamma([u_1]_i\sigma))\downarrow = \mathsf{f}(\delta_\gamma([u_1]_i\sigma)\downarrow)\downarrow = \delta_\gamma(v_j)\downarrow$. We have shown that $\delta_\gamma(v_j) = \delta_\gamma([u]_i\sigma\downarrow)$, thus by Lemma 6, $\delta_\gamma(v_j)$ is in normal form and which allows us to conclude.

Corollary 1. *Let $u, v \in \mathcal{T}(\Sigma_i \cup \Sigma_0, \mathcal{N} \cup \mathcal{X})$ for some $i \in \{1, \ldots, p\}$. Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. Assume that $fv(u) \cup fv(v) \subseteq \mathrm{dom}(\sigma_0{\downarrow})$, and $\sigma_0 \vDash \mathsf{test}_i([u]_i) \wedge \mathsf{test}_i([v]_i)$. Moreover, assume that u, v do not use names in $\mathcal{E}_\alpha \uplus \mathcal{E}_\beta$.*

$$[u]_i\sigma_0{\downarrow} = [v]_i\sigma_0{\downarrow} \Leftrightarrow \delta_\gamma([u]_i)\delta_\gamma(\sigma_0{\downarrow}){\downarrow} = \delta_\gamma([v]_i)\delta_\gamma(\sigma_0{\downarrow}){\downarrow}.$$

Proof. Thanks to Lemma 4, we have that

$$[u]_i\sigma_0{\downarrow} = [v]_i\sigma_0{\downarrow} \Leftrightarrow \delta_\gamma([u]_i\sigma_0{\downarrow}) = \delta_\gamma([v]_i\sigma_0{\downarrow}).$$

Thanks to Lemma 7, we have that:

- $\delta_\gamma([u]_i\sigma_0{\downarrow}) = \delta_\gamma([u]_i(\sigma_0{\downarrow})){\downarrow} = \delta_\gamma([u]_i)\delta_\gamma(\sigma_0{\downarrow}){\downarrow}$, and
- $\delta_\gamma([v]_i\sigma_0{\downarrow}) = \delta_\gamma([v]_i(\sigma_0{\downarrow})){\downarrow} = \delta_\gamma([v]_i)\delta_\gamma(\sigma_0{\downarrow}){\downarrow}$.

This allows us to conclude.

Lemma 8. *Let $u \in \mathcal{T}(\Sigma_i \cup \Sigma_0, \mathcal{N} \cup \mathcal{X})$ for some $i \in \{1, \ldots, p\}$. Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. Assume that $fv(u) \subseteq \mathrm{dom}(\sigma_0)$. Moreover, assume that u does not use names in $\mathcal{E}_\alpha \cup \mathcal{E}_\beta$. We have that :*

$$\sigma_0{\downarrow} \vDash \mathsf{test}_i([u]_i) \;\Leftrightarrow\; \delta_\gamma(\sigma_0{\downarrow}) \vDash \mathsf{test}_i(\delta_\gamma([u]_i))$$

Proof. To simplify the proof, we denote by σ the substitution $\sigma_0{\downarrow}$. We prove this result by induction on $|u|$:

Base case $|u| = 1$: In this case, we have that $u \in \mathcal{N} \cup \mathcal{X}$, and thus $[u]_i, \delta_\gamma([u]_i) \in \mathcal{N} \cup \mathcal{X}$. In such a case, we have that $\mathsf{test}_i([u]_i) = \mathsf{true}$ and $\mathsf{test}_i(\delta_\gamma([u]_i)) = \mathsf{true}$. Hence, the result trivially holds.

Inductive step $|u| > 1$, i.e. $u = \mathsf{f}(u_1, \ldots, u_n)$. We do a case analysis on f:

Case $\mathsf{f} \in \Sigma_i \cup \{\mathsf{pk}, \mathsf{vk}, \langle\ \rangle\}$: In this case, we have that $[u]_i = \mathsf{f}([u_1]_i, \ldots, [u_n]_i)$ and $\delta_\gamma([u]_i) = \mathsf{f}(\delta_\gamma([u_1]_i), \ldots, \delta_\gamma([u_n]_i))$. Thus, we deduce that $\mathsf{test}_i([u]_i) = \bigwedge_{j=1}^n \mathsf{test}_i([u_j]_i)$ and $\mathsf{test}_i(\delta_\gamma([u]_i)) = \bigwedge_{j=1}^n \mathsf{test}_i(\delta_\gamma([u_j]_i))$. By inductive hypothesis on $u_1, \ldots, u_n$, the result holds.

Case $\mathsf{f} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\}$: In this case, we have that:

- $[u]_i = \mathsf{f}(\mathsf{tag}_i([u_1]_i), [u_2]_i)$, and
- $\delta_\gamma([u]_i) = \mathsf{f}(\mathsf{tag}_i(\delta_\gamma([u_1]_i)), \delta_\gamma([u_2]_i))$.

Thus, we deduce that $\mathsf{test}_i([u]_i) = \mathsf{test}_i([u_1]_i) \wedge \mathsf{test}_i([u_2]_i)$ and $\mathsf{test}_i(\delta_\gamma([u]_i)) = \mathsf{test}_i(\delta_\gamma([u_1]_i)) \wedge \mathsf{test}_i(\delta_\gamma([u_2]_i))$. By inductive hypothesis on u_1, u_2, the result holds.

Case $\mathsf{f} = \mathsf{h}$: This case is analogous to de previous one and can be handled in a similar way.

Case $\mathsf{f} \in \{\mathsf{sdec}, \mathsf{adec}, \mathsf{check}\}$: In this case, we have that:

- $[u]_i = \mathsf{untag}_i(\mathsf{f}([u_1]_i, [u_2]_i))$, and

- $\delta_\gamma([u]_i) = \mathsf{untag}_i(\mathsf{f}(\delta_\gamma([u_1]_i), \delta_\gamma([u_2]_i)))$.

Thus, we deduce that $\mathsf{test}_i([u]_i)$ is the following formula:

$$\begin{array}{l}\mathsf{test}_i([u_1]_i) \wedge \mathsf{test}_i([u_2]_i) \\ \wedge \mathsf{tag}_i(\mathsf{untag}_i(\mathsf{f}([u_1]_i, [u_2]_i))) = \mathsf{f}([u_1]_i, [u_2]_i)\end{array}$$

and $\mathsf{test}_i(\delta_\gamma([u]_i))$ is the following formula:

$$\begin{array}{l}\mathsf{test}_i(\delta_\gamma([u_1]_i)) \wedge \mathsf{test}_i(\delta_\gamma([u_2]_i)) \wedge \\ \mathsf{tag}_i(\mathsf{untag}_i(\mathsf{f}(\delta_\gamma([u_1]_i), \delta_\gamma([u_2]_i)))) = \mathsf{f}(\delta_\gamma([u_1]_i), \delta_\gamma([u_2]_i))\end{array}$$

Whether we assume that $\sigma \vDash \mathsf{test}_i([u]_i)$ or $\delta_\gamma(\sigma) \vDash \mathsf{test}_i(\delta_\gamma([u]_i))$, we have by inductive hypothesis that $\sigma \vDash \mathsf{test}_i([u_k]_i)$ with $k \in \{1, 2\}$. Thus by Lemma 7, it implies that $\delta_\gamma([u_k]_i\sigma\!\downarrow) = \delta_\gamma([u_k]_i)\delta_\gamma(\sigma)\!\downarrow$ with $k \in \{1, 2\}$. We do a case analysis on f. We detail below the case where $\mathsf{f} = \mathsf{sdec}$. The cases where $\mathsf{f} = \mathsf{adec}$, and $\mathsf{f} = \mathsf{check}$ can be done in a similar way.

In such a case ($\mathsf{f} = \mathsf{sdec}$), we have that

$$\sigma \vDash \mathsf{tag}_i(\mathsf{untag}_i(\mathsf{f}([u_1]_i, [u_2]_i))) = \mathsf{f}([u_1]_i, [u_2]_i)$$

is equivalent to there exists v_1, v_2 s.t. $[u_2]_i\sigma\!\downarrow = v_2$ and $[u_1]_i\sigma\!\downarrow = \mathsf{senc}(\mathsf{tag}_i(v_1), v_2)$. But by Lemma 4, it is equivalent to $\delta_\gamma([u_1]_i\sigma\!\downarrow) = \mathsf{senc}(\mathsf{tag}_i(\delta_\gamma(v_1)), \delta_\gamma(v_2))$ and $\delta_\gamma([u_2]_i\sigma\!\downarrow) = \delta_\gamma(v_2)$. Thus, it is equivalent to:

- $\delta_\gamma([u_1]_i)\delta_\gamma(\sigma)\!\downarrow = \mathsf{senc}(\mathsf{tag}_i(\delta_\gamma(v_1)), \delta_\gamma(v_2))$, and
- $\delta_\gamma([u_2]_i)\delta_\gamma(\sigma)\!\downarrow = \delta_\gamma(v_2)$.

Hence it is equivalent to

$$\delta_\gamma(\sigma) \vDash \left(\begin{array}{l}\mathsf{tag}_i(\mathsf{untag}_i(\mathsf{f}(\delta_\gamma([u_1]_i), \delta_\gamma([u_2]_i)))) \\ = \mathsf{f}(\delta_\gamma([u_1]_i), \delta_\gamma([u_2]_i))\end{array}\right)$$

Case $\mathsf{f} \in \{\mathsf{proj}_1, \mathsf{proj}_2\}$*:* In such a case, we have that $[u]_i = \mathsf{f}([u_1]_i)$ and $\delta_\gamma([u]_i) = \mathsf{f}(\delta_\gamma([u_1]_i))$. Thus, we deduce that $\mathsf{test}_i([u]_i)$ is the following formula:

$$\mathsf{test}_i([u_1]_i) \wedge \langle \mathsf{proj}_1([u_1]_i), \mathsf{proj}_2([u_1]_i)\rangle = [u_1]_i$$

and $\mathsf{test}_i(\delta_\gamma([u]_i))$ is the following formula:

$$\begin{array}{l}\mathsf{test}_i(\delta_\gamma([u_1]_i)) \wedge \\ \langle \mathsf{proj}_1(\delta_\gamma([u_1]_i)), \mathsf{proj}_2(\delta_\gamma([u_1]_i))\rangle = \delta_\gamma([u_1]_i)\end{array}$$

Whether we assume that $\sigma \vDash \mathsf{test}_i([u]_i)$ or $\delta_\gamma(\sigma) \vDash \mathsf{test}_i(\delta_\gamma([u]_i))$, we have by inductive hypothesis that $\sigma \vDash \mathsf{test}_i([u_1]_i)$. Thus by Lemma 7, it implies that $\delta_\gamma([u_1]_i\sigma\!\downarrow) = \delta_\gamma([u_1]_i)\delta_\gamma(\sigma)\!\downarrow$.

Actually $\sigma \vDash \langle \mathsf{proj}_1([u_1]_i), \mathsf{proj}_2([u_1]_i)\rangle = [u_1]_i$ is equivalent to there exist v_1, v_2 such that $[u_1]_i\sigma\!\downarrow = \langle v_1, v_2\rangle$, which is, thanks to Lemma 4, equivalent to $\delta_\gamma([u_1]_i\sigma\!\downarrow) = \langle \delta_\gamma(v_1), \delta_\gamma(v_2)\rangle$.

We have shown that this is equivalent to

$$\delta_\gamma([u_1]_i)\delta_\gamma(\sigma)\!\downarrow = \langle \delta_\gamma(v_1), \delta_\gamma(v_2)\rangle$$

Thus, we conclude that $\sigma \vDash \langle \mathsf{proj}_1([u_1]_i), \mathsf{proj}_2([u_1]_i)\rangle = [u_1]_i$ is equivalent $\delta_\gamma(\sigma) \vDash \langle \mathsf{proj}_1(\delta_\gamma([u_1]_i)), \mathsf{proj}_2(\delta_\gamma([u_1]_i))\rangle = \delta_\gamma([u_1]_i)$.

For a term u that does not contain any tag, we defined a way to construct a term that is properly tagged (*i.e.* $[u]_i$). Hence, for a term properly tagged, we would never have $\mathsf{senc}(n, k)$ where n and k are both nonces, for example. Instead, we would have $\mathsf{senc}(\mathsf{tag}_i(n), k)$. However, even if we can force the processes to properly tag their terms, we do not have any control on what the intruder can build. Typically, if the intruder is able to deduce n and k, he is allowed to send to a process the term $\mathsf{senc}(n, k)$. Thus, we want to define the notion of *flawed tagged term.*

Definition 17. *Let u be a ground term in normal form. Consider γ and γ' such that $\{\gamma, \gamma'\} = \{\alpha, \beta\}$. We define the flawed subterms of u w.r.t. γ, denoted* $\mathsf{Flawed}^\gamma(u)$*, as follows:*

$$\mathsf{Flawed}^\gamma(u) \overset{\mathsf{def}}{=} \left\{ v \in st(u) \,\middle|\, \begin{array}{l} \mathsf{tagroot}(v) \in \{0\} \cup \gamma' \textit{ and} \\ \mathsf{root}(v) \notin \{\mathsf{pk}, \mathsf{vk}, \langle\,\rangle\} \end{array} \right\}$$

We define the flawed subterms of u, denoted $\mathsf{Flawed}(u)$*, as the set* $\mathsf{Flawed}(u) = \mathsf{Flawed}^\alpha(u) \cap \mathsf{Flawed}^\beta(u)$

Lemma 9. *Let $u \in \mathcal{T}(\Sigma_i \cup \Sigma_0, \mathcal{N} \cup \mathcal{X})$ for some $i \in \{1, \ldots, p\}$. Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. Let γ' such that $\gamma' \in \{\alpha, \beta\} \smallsetminus \gamma$. Let σ be a ground substitution in normal form such that $fv(u) \subseteq \mathrm{dom}(\sigma)$.*

If $\sigma \vDash \mathsf{test}_i([u]_i)$ then for all $t \in \mathsf{Flawed}^\gamma([u]_i\sigma\!\downarrow)$, there exists $x \in fv([u]_i)$ such that $t \in \mathsf{Flawed}^\gamma(x\sigma)$.

Proof. We prove the result by induction on $|u|$.

Base case $|u| = 1$: In this case, we have that $u \in \mathcal{X} \cup \mathcal{N}$ and so $[u]_i = u$. If $u \in \mathcal{N}$, then $u\sigma$ and $[u]_i\sigma\!\downarrow$ are both in $\mathcal{N}$, which means that $\mathsf{Flawed}^\gamma([u]_i\sigma\!\downarrow) = \emptyset$. Thus, the result holds. Otherwise, we have that $u \in \mathcal{X}$ and so $[u]_i = u \in \mathrm{dom}(\sigma)$ which means that the result trivially holds.

Inductive step $|u| > 1$, i.e. $u = \mathsf{f}(u_1, \ldots, u_n)$. We do a case analysis on f.

Case $\mathsf{f} \in \Sigma_i$: In this case, $[u]_i = \mathsf{f}([u_1]_i, \ldots, [u_n]_i)$ and $[u]_i\sigma\!\downarrow = \mathsf{f}([u_1]_i\sigma\!\downarrow, \ldots, [u_n]_i\sigma\!\downarrow)\!\downarrow$. By definition, we know that for all $t \in \mathsf{Flawed}^\gamma([u]_i\sigma\!\downarrow)$, $\mathsf{root}(t) \notin \Sigma_\gamma$. Thus, thanks to Lemma 2, for all $t \in \mathsf{Flawed}^\gamma([u]_i\sigma\!\downarrow)$, there exists $k \in \{1, \ldots, n\}$ such that $t \in st([u_k]_i\sigma\!\downarrow)$. By hypothesis, $\sigma \vDash \mathsf{test}_i([u]_i)$ and so $\sigma \vDash \mathsf{test}_i([u_k]_i)$. Thus, by inductive hypothesis, we know that there exists $x \in fv([u_k]_i)$ such that $t \in st(x\sigma)$. Since $fv([u_k]_i) \subseteq fv([u]_i)$, we can conclude.

Case $\mathsf{f} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\}$*:* In such a case, $[u]_i = \mathsf{f}(\mathsf{tag}_i([u_1]_i), [u_2]_i)$ and $[u]_i\sigma\downarrow = \mathsf{f}(\mathsf{tag}_i([u_1]_i\sigma\downarrow), [u_2]_i\sigma\downarrow)$. Moreover, $\sigma \vDash \mathsf{test}_i([u]_i)$ implies that $\sigma \vDash \mathsf{test}_i([u_k]_i)$, with $k \in \{1,2\}$. Since $\mathsf{tagroot}([u]_i\sigma\downarrow) = i$, then we deduce that :

$$\mathsf{Flawed}^\gamma([u]_i\sigma\downarrow) = \mathsf{Flawed}^\gamma([u_1]_i\sigma\downarrow) \cup \mathsf{Flawed}^\gamma([u_2]_i\sigma\downarrow)$$

Thanks to our inductive hypothesis on u_1 and u_2, the result holds.

Case $\mathsf{f} = \mathsf{h}$*:* This case is analogous to the previous one and can be handled in a similar way.

Case $\mathsf{f} = \langle\ \rangle$*:* In this case, we have that $[u]_i = \mathsf{f}([u_1]_i, [u_2]_i)$, and $[u]_i\sigma\downarrow = \mathsf{f}([u_1]_i\sigma\downarrow, [u_2]_i\sigma\downarrow)$. Moreover, $\sigma \vDash \mathsf{test}_i([u]_i)$ implies that $\sigma \vDash \mathsf{test}_i([u_k]_i)$ with $k \in \{1,2\}$. By definition, since $\mathsf{root}([u]_i\sigma\downarrow) = \langle\ \rangle$, we have that $\mathsf{Flawed}^\gamma([u]_i\sigma\downarrow) = \mathsf{Flawed}^\gamma([u_1]_i\sigma\downarrow) \cup \mathsf{Flawed}^\gamma([u_2]_i\sigma\downarrow)$. Applying our inductive hypothesis on u_1 and u_2, we conclude.

Case $\mathsf{f} = \{\mathsf{vk}, \mathsf{pk}\}$*:* In this case, we have $u = \mathsf{f}(v)$ with $v \in \mathcal{N} \cup \mathcal{X}$. Thus $[u]_i = u$ and so by definition, $\mathsf{Flawed}^\gamma(u\sigma\downarrow) = \emptyset$. Thus, the result trivially holds.

Case $\mathsf{f} \in \{\mathsf{sdec}, \mathsf{adec}, \mathsf{check}\}$*:* In this case, we have that $[u]_i = \mathsf{untag}_i(\mathsf{f}([u_1]_i, [u_2]_i))$ and

$$\begin{aligned}\mathsf{test}_i([u]_i) = \mathsf{test}_i([u_1]_i) \wedge \mathsf{test}_i([u_2]_i) \wedge \\ \mathsf{tag}_i([u]_i) = \mathsf{f}([u_1]_i, [u_2]_i).\end{aligned}$$

By hypothesis, we know that $\sigma \vDash \mathsf{test}_i([u]_i)$ and more specifically $\mathsf{tag}_i([u]_i)\sigma\downarrow = \mathsf{f}([u_1]_i, [u_2]_i)\sigma\downarrow$. It implies that there exist v_1, v_2 such that $[u_1]_i\sigma\downarrow = \mathsf{g}(\mathsf{tag}_i(v_1), v_2)$ and $[u]_i\sigma\downarrow = v_1$, with $\mathsf{g} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\}$. Thus, for all $t \in \mathsf{Flawed}^\gamma([u]_i\sigma\downarrow)$, $t \in \mathsf{Flawed}^\gamma([u_1]_i\sigma\downarrow)$. Since $\sigma \vDash \mathsf{test}_i([u_1]_i)$, the result holds by inductive hypothesis.

Case $\mathsf{f} = \mathsf{proj}_j$, $j \in \{1,2\}$*:* We have that $[u]_i = \mathsf{f}([u_1]_i)$ and $\mathsf{test}_i([u]_i) = \mathsf{test}_i([u_1]_i) \wedge \langle \mathsf{proj}_1([u_1]_i), \mathsf{proj}_2([u_1]_i)\rangle = [u_1]_i$. Hence, $\sigma \vDash \mathsf{test}_i([u]_i)$ implies that there exist v_1, v_2 such that $[u_1]_i\sigma\downarrow = \langle v_1, v_2\rangle$ and $[u]_i\sigma\downarrow = v_j$. Thus, for all $t \in \mathsf{Flawed}^\gamma([u]_i\sigma\downarrow)$, $t \in \mathsf{Flawed}^\gamma([u_1]_i\sigma\downarrow)$. Since $\sigma \vDash \mathsf{test}_i([u_1]_i)$, our inductive hypothesis allows us to conclude.

Corollary 2. *Let $u \in \mathcal{T}(\Sigma_i \cup \Sigma_0, \mathcal{N} \cup \mathcal{X})$ for some $i \in \{1, \dots, p\}$. Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. Let σ be a ground substitution in normal form such that $fv(u) \subseteq$* $\mathrm{dom}(\sigma)$.

If $\sigma \vDash \mathsf{test}_i([u]_i)$ then for all $t \in \mathsf{Flawed}([u]_i\sigma\downarrow)$, there exists $x \in fv([u]_i)$ such that $t \in \mathsf{Flawed}(x\sigma)$.

Corollary 3. *Let $u \in \mathcal{T}(\Sigma_i \cup \Sigma_0, \mathcal{N} \cup \mathcal{X})$ for some $i \in \{1, \dots, p\}$. Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. Assume that $fv(u) \subseteq \mathrm{dom}(\sigma_0)$. Moreover, assume that u does not use names in $\mathcal{E}_\alpha \cup \mathcal{E}_\beta$. If $\delta_\gamma(\sigma_0\downarrow) \vDash \mathsf{test}_i(\delta_\gamma([u]_i))$, then for all $t \in \mathsf{Flawed}^\gamma(\delta_\gamma([u]_i)\delta_\gamma(\sigma_0\downarrow)\downarrow)$, there exists $x \in fv(\delta_\gamma([u]_i))$ such that $t \in \mathsf{Flawed}^\gamma(x\delta_\gamma(\sigma_0\downarrow))$.*

Proof. By Lemma 6, we deduce that $\delta_\gamma(\sigma_0\!\downarrow)$ is a substitution in normal form. Moreover, since $u \in \mathcal{T}(\Sigma_i \cup \Sigma_0, \mathcal{N} \cup \mathcal{X})$ and by definition of δ_γ, and $[\ ]_i$, we deduce that there exists $v \in \mathcal{T}(\Sigma_i \cup \Sigma_0, \mathcal{N} \cup \mathcal{X})$ such that $[v]_i = \delta_\gamma([u]_i)$. By application of Lemma 9, we deduce that for all $t \in \mathsf{Flawed}^\gamma([v]_i\delta_\gamma(\sigma_0\!\downarrow)\!\downarrow)$, there exists $x \in fv([v]_i)$ such that $t \in st(x\delta_\gamma(\sigma_0\!\downarrow))$. Hence, we conclude that for all $t \in \mathsf{Flawed}^\gamma(\delta_\gamma([u]_i)\delta_\gamma(\sigma_0\!\downarrow)\!\downarrow)$, there exists $x \in fv(\delta_\gamma([u]_i))$ such that $t \in st(x\delta_\gamma(\sigma_0\!\downarrow))$.

Definition 18. *Let $u \in \mathcal{T}(\Sigma, \mathcal{N} \cup \mathcal{X})$. The α-factors (resp. β-factors) of u, denoted $Fct_\alpha(u)$, are the maximal syntactic subterms of u that are also in $\mathsf{Flawed}^\alpha(u)$ (resp. $\mathsf{Flawed}^\beta(u)$).*

Lemma 10. *Let $u \in \mathcal{T}(\Sigma_i \cup \Sigma_0, \mathcal{N} \cup \mathcal{X})$ for some $i \in \{1, \ldots, p\}$. Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. Let σ be a ground substitution in normal form such that $fv(u) \subseteq$* $\mathrm{dom}(\sigma)$.

If $\sigma \vDash \mathsf{test}_i([u]_i)$ then

- *either $[u]_i\sigma\!\downarrow\, \in Fct_\gamma([u]_i\sigma)$,*
- *otherwise $Fct_\gamma([u]_i\sigma\!\downarrow) \subseteq Fct_\gamma([u]_i\sigma)$*

Proof. We prove the result by induction on $|u|$.

Base case $|u| = 1$: In this case, we have that $u \in \mathcal{X} \cup \mathcal{N}$ and so $[u]_i = u$. If $u \in \mathcal{N}$, then $u\sigma$ and $[u]_i\sigma\!\downarrow$ are both in $\mathcal{N}$, which means that $Fct_\gamma([u]_i\sigma) = \emptyset$ and $Fct_\gamma([u]_i\sigma\!\downarrow) = \emptyset$. Thus, the result holds. Otherwise, we have that $u \in \mathcal{X}$ and so $[u]_i = u$. But σ is in normal form hence $[u]_i\sigma\!\downarrow\, = [u]_i\sigma$. Thus, $Fct_\alpha([u]_i\sigma\!\downarrow) = Fct_\alpha([u]_i\sigma\!\downarrow)$ and so the result holds.

Inductive step $|u| > 1$, i.e. $u = \mathsf{f}(u_1, \ldots, u_n)$. We do a case analysis on f.

Case $\mathsf{f} \in \Sigma_i$: In this case, $[u]_i = \mathsf{f}([u_1]_i, \ldots, [u_n]_i)$ and $[u]_i\sigma\!\downarrow\, = \mathsf{f}([u_1]_i\sigma\!\downarrow, \ldots, [u_n]_i\sigma\!\downarrow)\!\downarrow$.

By definition, we know that for all $t \in Fct_\gamma[u]_i\sigma\!\downarrow$, $\mathsf{root}(t) \not\in \Sigma_\gamma$. Thus, thanks to Lemma 2, for all $t \in Fct_\gamma[u]_i\sigma\!\downarrow$, there exists $k \in \{1, \ldots, n\}$ such that $t \in Fct_\gamma([u_k]_i\sigma\!\downarrow)$. By hypothesis, $\sigma \vDash \mathsf{test}_i([u]_i)$ and so $\sigma \vDash \mathsf{test}_i([u_k]_i)$. Thus, by inductive hypothesis, we know that

- either $[u_k]_i\sigma\!\downarrow\, \in Fct_\gamma([u_k]_i\sigma)$,
- otherwise $Fct_\gamma([u_k]_i\sigma\!\downarrow) \subseteq Fct_\gamma([u_k]_i\sigma)$

Thus, if $[u_k]_i\sigma\!\downarrow\, \in Fct_\gamma([u_k]_i\sigma)$ then it means that $t = [u_k]_i\sigma\!\downarrow$ and so $t \in Fct_\gamma([u_k]_i\sigma)$ (otherwise it contradicts the notion of maximal subterm). Thus in both cases, we obtain that $t \in Fct_\gamma([u_k]_i\sigma)$. Since $Fct_\gamma([u_k]_i\sigma) \subseteq Fct_\gamma([u]_i\gamma)$ then we deduce that $t \in Fct_\gamma([u]_i\gamma)$ hence the result holds.

Case $\mathsf{f} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\}$: In such a case, $[u]_i = \mathsf{f}(\mathsf{tag}_i([u_1]_i), [u_2]_i)$ and $[u]_i\sigma\!\downarrow\, = \mathsf{f}(\mathsf{tag}_i([u_1]_i\sigma\!\downarrow), [u_2]_i\sigma\!\downarrow)$. Moreover, $\sigma \vDash \mathsf{test}_i([u]_i)$ implies that $\sigma \vDash \mathsf{test}_i([u_k]_i)$, with $k \in \{1, 2\}$. Since $\mathsf{tagroot}([u]_i\sigma\!\downarrow) = i$, then we deduce that :

$$Fct_\gamma([u]_i\sigma\!\downarrow) = Fct_\gamma([u_1]_i\sigma\!\downarrow) \cup Fct_\gamma([u_2]_i\sigma\!\downarrow)$$

Thanks to our inductive hypothesis on u_1 and u_2, the result holds.

Case $\mathsf{f} = \mathsf{h}$*:* This case is analogous to the previous one and can be handled in a similar way.

Case $\mathsf{f} = \langle\ \rangle$*:* In this case, we have that $[u]_i = \mathsf{f}([u_1]_i, [u_2]_i)$, and $[u]_i\sigma\downarrow = \mathsf{f}([u_1]_i\sigma\downarrow, [u_2]_i\sigma\downarrow)$. Moreover, $\sigma \models \mathsf{test}_i([u]_i)$ implies that $\sigma \models \mathsf{test}_i([u_k]_i)$ with $k \in \{1, 2\}$. By definition, since $\mathsf{root}([u]_i\sigma\downarrow) = \langle\ \rangle$, we have that $Fct_\gamma([u]_i\sigma\downarrow) = Fct_\gamma([u_1]_i\sigma\downarrow) \cup Fct_\gamma([u_2]_i\sigma\downarrow)$. Applying our inductive hypothesis on u_1 and u_2, we conclude.

Case $\mathsf{f} = \{\mathsf{vk}, \mathsf{pk}\}$*:* In this case, we have $u = \mathsf{f}(v)$ with $v \in \mathcal{N} \cup \mathcal{X}$. Thus $[u]_i = u$ and so by definition, $Fct_\gamma(u\sigma\downarrow) = \emptyset$. Thus, the result trivially holds.

Case $\mathsf{f} \in \{\mathsf{sdec}, \mathsf{adec}, \mathsf{check}\}$*:* In this case, we have that $[u]_i = \mathsf{untag}_i(\mathsf{f}([u_1]_i, [u_2]_i))$ and

$$\begin{aligned}\mathsf{test}_i([u]_i) = \mathsf{test}_i([u_1]_i) \wedge \mathsf{test}_i([u_2]_i) \wedge \\ \mathsf{tag}_i([u]_i) = \mathsf{f}([u_1]_i, [u_2]_i).\end{aligned}$$

By hypothesis, we know that $\sigma \models \mathsf{test}_i([u]_i)$ and more specifically $\mathsf{tag}_i([u]_i)\sigma\downarrow = \mathsf{f}([u_1]_i, [u_2]_i)\sigma\downarrow$. It implies that there exists v_1, v_2 such that $[u_1]_i\sigma\downarrow = \mathsf{g}(\mathsf{tag}_i(v_1), v_2)$ and $[u]_i\sigma\downarrow = v_1$, with $\mathsf{g} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\}$. Thus, for all $t \in Fct_\gamma([u]_i\sigma\downarrow)$, $t \in Fct_\gamma([u_1]_i\sigma\downarrow)$. Since $\sigma \models \mathsf{test}_i([u_1]_i)$, the result holds by inductive hypothesis.

Case $\mathsf{f} = \mathsf{proj}_j$*,* $j \in \{1, 2\}$*:* We have that $[u]_i = \mathsf{f}([u_1]_i)$ and $\mathsf{test}_i([u]_i) = \mathsf{test}_i([u_1]_i) \wedge \langle \mathsf{proj}_1([u_1]_i), \mathsf{proj}_2([u_1]_i)\rangle = [u_1]_i$. Hence, $\sigma \models \mathsf{test}_i([u]_i)$ implies that there exist v_1, v_2 such that $[u_1]_i\sigma\downarrow = \langle v_1, v_2\rangle$ and $[u]_i\sigma\downarrow = v_j$. Thus, for all $t \in Fct_\gamma([u]_i\sigma\downarrow)$, $t \in Fct_\gamma([u_1]_i\sigma\downarrow)$. Since $\sigma \models \mathsf{test}_i([u_1]_i)$, our inductive hypothesis allows us to conclude.

E.5 Frame of a tagged process

In this subsection, we will state and prove the lemmas regarding frames and static equivalence. Let $\nu\mathcal{E}.\Phi$ be a frame such that:

$$\Phi = \{w_1 \rhd u_1, \dots, w_n \rhd u_n\}.$$

Let M be a recipe, *i.e.* a term such that $fv(M) \subseteq \mathrm{dom}(\Phi)$ and $fn(M) \cap \mathcal{E} = \emptyset$, we define the measure μ as follows:

$$\mu(M) = (i_{\mathsf{max}}, |M|)$$

where $i_{\mathsf{max}} \in \{1, \dots, n\}$ is the maximal indice i such that $w_i \in fv(M)$, and $|M|$ denotes the size of the term M, *i.e.* the number of symbols that occur in M.

We have that $\mu(M_1) \overset{\mathsf{def}}{=} (i_1, s_1) < \mu(M_2) \overset{\mathsf{def}}{=} (i_2, s_2)$ when either $i_1 < i_2$; or $i_1 = i_2$ and $s_1 < s_2$.

Once again, we denote by $z_1^\alpha, \dots, z_k^\alpha$ and $z_1^\beta, \dots, z_\ell^\beta$ the assignment variables of the extended processes that we are considering.

Definition 19. *Let $(\mathcal{E};\mathcal{P};\Phi;\sigma)$ be an extended process, $\prec$ be a total order on* $\mathrm{dom}(\Phi)\cup\mathrm{dom}(\sigma)$ *and col be a mapping from* $\mathrm{dom}(\Phi)\cup\mathrm{dom}(\sigma)$ *to* $\{1,\ldots,p\}$. *We say that $(\mathcal{E};\mathcal{P};\Phi;\sigma)$ is a* derived well-tagged extended process *w.r.t. $\prec$ and col if for every $x\in\mathrm{dom}(\Phi)$ (resp. $x\in\mathrm{dom}(\sigma)$), there exists $\{\gamma,\gamma'\}=\{\alpha,\beta\}$ such that one of the following condition is satisfied:*

1. *there exist v and $i=col(x)\in\gamma$ such that $u=[v]_i\sigma$, $\sigma\vDash\mathsf{test}_i([v]_i)$, and for all $z\in fv(v)$, $z\prec x$ and either $col(z)\in\gamma$ or there exists j such that $z=z_j^{\gamma'}$; or*
2. *there exists M such that $fv(M)\subseteq\mathrm{dom}(\Phi)\cap\{z\mid z\prec x\}$, $fn(M)\cap\mathcal{E}=\emptyset$ and $M\Phi=u$.*

where $u=x\Phi$ (resp. $u=x\sigma$).

In the case of variables instantiated through an output, and or an internal communication, it will be the first item that needs to hold; while in the case of variables intantiated through inputs on public channels it is the second item that needs to hold. Intuitively, the order $\prec$ on $\mathrm{dom}(\Phi)\cup\mathrm{dom}(\sigma)$ corresponds to the order in which the variables in $\mathrm{dom}(\Phi)\cup\mathrm{dom}(\sigma)$ have been introduced along the execution. In particular, we have that $w_1\prec w_2\prec\ldots\prec w_n$ where $\mathrm{dom}(\Phi)=\{w_1,\ldots,w_n\}$. In the following, we sometimes simply say that $(\mathcal{E};\mathcal{P};\Phi;\sigma)$ is a derived well-tagged extended process.

Lemma 11. *Let $(\mathcal{E};\mathcal{P};\Phi;\sigma)$ be a derived well-tagged extended process w.r.t $\prec$ and col. Let $x\in\mathrm{dom}(\Phi)$ (resp. $x\in\mathrm{dom}(\sigma)$) and $t\in\mathsf{Flawed}(x\Phi\downarrow)$ (resp. $t\in\mathsf{Flawed}(x\sigma\downarrow)$). We have that there exists M such that $fv(M)\subseteq\mathrm{dom}(\Phi)\cap\{z\mid z\prec x\}$, $fn(M)\cap\mathcal{E}=\emptyset$ and $t\in\mathsf{Flawed}(M\Phi\downarrow)$.*

Proof. We prove this result by induction on $\mathrm{dom}(\Phi)\cup\mathrm{dom}(\sigma)$ with the order $\prec$.
Base case $u=x\sigma$ or $u=x\Phi$ with $x\prec z$ for any $z\in\mathrm{dom}(\Phi)\cup\mathrm{dom}(\sigma)$. Assume $t\in\mathsf{Flawed}(u\downarrow)$. By definition of a derived well-tagged extended process w.r.t $\prec$ and col, one of the following condition is satisfied:

1. There exist v and $i=col(x)$ such that $u=[v]_i\sigma$, $\sigma\vDash\mathsf{test}_i([v]_i)$, and $z\prec x$ for any $z\in fv(v)$. Since $u=[v]_i\sigma$ and $\sigma\vDash\mathsf{test}_i([v]_i)$, we can apply Lemma 9 to v and $\sigma\downarrow$. Thus, we have that there exists $z\in fv([v]_i)$ such that $t\in\mathsf{Flawed}(z\sigma\downarrow)$. However, since x is mimimal w.r.t. $\prec$, we know that $fv(v)=\emptyset$. Hence, we obtain a contradiction. This case is impossible.
2. There exists M such that $fv(M)\subseteq\mathrm{dom}(\Phi)\cap\{z\mid z\prec x\}$, $fn(M)\cap\mathcal{E}=\emptyset$, and $M\Phi=u$. Thus, we have that $M\Phi\downarrow=u\downarrow$, and we have that $t\in\mathsf{Flawed}(M\Phi\downarrow)$.

Inductive case $u=x\sigma$ or $u=x\Phi$. Assume $t\in\mathsf{Flawed}(u\downarrow)$. By definition of a derived well-tagged extended process w.r.t $\prec$ and col, one of the following condition is satisfied:

1. There exist v and $i=col(x)$ such that $u=[v]_i\sigma$, $\sigma\vDash\mathsf{test}_i([v]_i)$, and $z\prec x$ for any $z\in fv(v)$. Since $u=[v]_i\sigma$ and $\sigma\vDash\mathsf{test}_i([v]_i)$, we can apply Lemma 9 to v and $\sigma\downarrow$. Thus, we have that there exists $z\in fv([v]_i)$ such that $t\in\mathsf{Flawed}(z\sigma\downarrow)$, and we have that $z\prec x$. Hence, we conclude by applying our induction hypothesis.

2. There exists M such that $fv(M) \subseteq \mathrm{dom}(\Phi) \cap \{z \mid z \prec x\}$, $fn(M) \cap \mathcal{E} = \emptyset$, and $M\Phi = u$. Thus, we have that $M\Phi\!\downarrow = u\!\downarrow$, and we have that $t \in \mathsf{Flawed}(M\Phi\!\downarrow)$.

This allows us to conclude.

Lemma 12. *Let $(\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ be a derived well-tagged extended process w.r.t $\prec$ and col. Let $\{\gamma, \gamma'\} = \{\alpha, \beta\}$. Let $x \in \mathrm{dom}(\Phi)$ (resp. $x \in \mathrm{dom}(\sigma)$) such that $col(x) \in \gamma$. Let $u = x\Phi$ (resp. $u = x\sigma$). Let $t \in Fct_\gamma(u\!\downarrow)$. We have that*

- *either there exists M such that $fv(M) \subseteq \mathrm{dom}(\Phi) \cap \{z \mid z \prec x\}$, $fn(M) \cap \mathcal{E} = \emptyset$ and $t \in Fct_\gamma(M\Phi\!\downarrow)$;*
- *otherwise there exists j such that $z_j^{\gamma'} \prec x$ and $z_j^{\gamma'}\sigma\!\downarrow = t$.*

Proof. We prove this result by induction on $\mathrm{dom}(\Phi) \cup \mathrm{dom}(\sigma)$ with the order $\prec$.

Base case $u = x\sigma$ or $u = x\Phi$ with $x \prec z$ for any $z \in \mathrm{dom}(\Phi) \cup \mathrm{dom}(\sigma)$. Let $t \in Fct_\gamma(u\!\downarrow)$ and $col(x) \in \gamma$ with $\gamma \in \{\alpha, \beta\}$. By definition of a derived well-tagged extended process w.r.t $\prec$ and col, one of the following condition is satisfied:

1. There exist v and $i = col(x)$ such that $u = [v]_i\sigma$, $\sigma \vDash \mathsf{test}_i([v]_i)$, and $z \prec x$ for any $z \in fv(v)$. Since x is minimal by $\prec$ then $fv(v) = \emptyset$. Hence $u = [v]_i$. Thus we deduce that $Fct_\gamma(u\!\downarrow) = \emptyset$. Hence there is a contradiction with $t \in Fct_\gamma(u\!\downarrow)$ and so this condition cannot be satisfied.
2. There exists M such that $fv(M) \subseteq \mathrm{dom}(\Phi) \cap \{z \mid z \prec x\}$, $fn(M) \cap \mathcal{E} = \emptyset$, and $M\Phi = u$. Thus, we have that $M\Phi\!\downarrow = u\!\downarrow$ and so the result holds.

Inductive case $u = x\sigma$ or $u = x\Phi$. Assume $t \in Fct_\gamma(u\!\downarrow)$ and $col(x) \in \gamma$. By definition of a derived well-tagged extended process w.r.t $\prec$ and col, one of the following condition is satisfied:

1. There exist v and $i = col(x) \in \gamma$ such that $u = [v]_i\sigma$, $\sigma \vDash \mathsf{test}_i([v]_i)$, and for all $z \in fv(v)$, $z \prec x$ and either $col(z) \in \gamma$ or there exists j such that $z = z_j^{\gamma'}$. Since $u = [v]_i\sigma$ and $\sigma \vDash \mathsf{test}_i([v]_i)$, we can apply Lemma 10 to v and $\sigma\!\downarrow$. Thus we have that $t \in Fct_\gamma([v]_i(\sigma\!\downarrow)$. In such a case, it means that there exists $z \in fv(v)$ with $z \prec x$ such that $t \in Fct_\gamma(z\sigma\!\downarrow)$ and one of the two conditions is satisfied:
   - $col(z) \in \gamma$: In such a case, we can apply our inductive hypothesis on t and z and so the result holds.
   - there exists j such that $z = z_j^{\gamma'}$: Otherwise, we know by hypothesis that $z^{\gamma'}\sigma\!\downarrow \in \mathcal{N}$ or $Fct_\gamma(z^{\gamma'}\sigma\!\downarrow) = \{z^{\gamma'}\sigma\!\downarrow\}$. Since $t \in Fct_\gamma(z\sigma\!\downarrow)$, we deduce that $z^{\gamma'}\sigma\!\downarrow \notin \mathcal{N}$ and so $Fct_\gamma(z^{\gamma'}\sigma\!\downarrow) = \{z^{\gamma'}\sigma\!\downarrow\}$. But this implies that $t = z\sigma\!\downarrow$. Hence the result holds.
2. There exists M such that $fv(M) \subseteq \mathrm{dom}(\Phi) \cap \{z \mid z \prec x\}$, $fn(M) \cap \mathcal{E} = \emptyset$, and $M\Phi = u$. Thus, we have that $M\Phi\!\downarrow = u\!\downarrow$, and we have that $t \in Fct_\gamma(M\Phi\!\downarrow)$.

This allows us to conclude.

Lemma 13. *Let $(\mathcal{E};\mathcal{P};\Phi;\sigma)$ be a derived well-tagged extended process. Let M be a term such that $fn(M) \cap \mathcal{E} = \emptyset$ and $fv(M) \subseteq \mathrm{dom}(\Phi)$. Let $\mathsf{f}(t_1,\ldots,t_m) \in \mathsf{Flawed}(M\Phi\downarrow)$. There exists $M_1,\ldots,M_m$ such that $fv(M_k) \subseteq \mathrm{dom}(\Phi)$, $fn(M_k) \cap \mathcal{E} = \emptyset$, $M_k\Phi\downarrow = t_k$, and $\mu(M_k) < \mu(M)$, for all $k \in \{1,\ldots,m\}$.*

Proof. We prove this result by induction on $\mu(M)$.

Base case $\mu(M) = (j,1)$: In this case, either we have that $M \in \mathcal{N}$ or $M = w_j$. If $M \in \mathcal{N}$, then we have $M\Phi\downarrow = M \in \mathcal{N}$ and $\mathsf{Flawed}(M\Phi\downarrow) = \emptyset$. Thus the result holds. If $M = w_j$ then, by Lemma 11, $\mathsf{f}(t_1,\ldots,t_m) \in \mathsf{Flawed}(w_j\Phi\downarrow)$ implies that there exists M' such that:

– $fv(M') \subseteq \{w_1,\ldots,w_{j-1}\}$,
– $fn(M) \cap \mathcal{E} = \emptyset$, and
– $\mathsf{f}(t_1,\ldots,t_m) \in \mathsf{Flawed}(M'\Phi\downarrow)$.

Since $\mu(M') < \mu(M)$, thanks to our inductive hypothesis, we deduce that there exist $M_1,\ldots,M_m$ such that for each $k \in \{1,\ldots,m\}$, we have that: $fv(M_k) \subseteq \mathrm{dom}(\Phi)$, $fn(M_k) \cap \mathcal{E} = \emptyset$, $M_k\Phi\downarrow = t_k$, and $\mu(M_k) < \mu(M') < \mu(M)$.

Inductive step $\mu(M) > (j,1)$: In such a case, we have that $M = \mathsf{f}(M_1,\ldots,M_n)$. Let $t = \mathsf{g}(t_1,\ldots,t_m) \in \mathsf{Flawed}(M\Phi\downarrow)$. We do a case analysis on f.

Case $\mathsf{f} \in \Sigma_i \cup \Sigma_{\mathsf{tag}_i}$ for some $i \in \{1,\ldots,p\}$: In such a case, $M\Phi\downarrow = \mathsf{f}(M_1\Phi\downarrow,\ldots,M_n\Phi\downarrow)\downarrow$. By definition, we know that for all $t \in \mathsf{Flawed}(M\Phi\downarrow)$, we have that $\mathsf{root}(t) \notin \Sigma_i \cup \Sigma_{\mathsf{tag}_i}$. Thus, thanks to Lemma 2, we deduce that

$$\mathsf{Flawed}(M\Phi\downarrow) \subseteq \mathsf{Flawed}(M_1\Phi\downarrow) \cup \ldots \cup \mathsf{Flawed}(M_n\Phi\downarrow).$$

Since $\mu(M_k) < \mu(M)$ for any $k \in \{1,\ldots,n\}$, thanks to our inductive hypothesis, we know that there exists $M'_1,\ldots,M'_m$ such that $fv(M'_j) \subseteq \mathrm{dom}(\Phi)$, $fn(M'_j) \cap \mathcal{E} = \emptyset$, $M'_j\Phi\downarrow = t_i$ and $\mu(M'_j) < \mu(M_k) < \mu(M)$, for $j \in \{1,\ldots,m\}$. Hence the result holds.

Case $\mathsf{f} = \langle\ \rangle$: In such a case, $M\Phi\downarrow = \mathsf{f}(M_1\Phi\downarrow, M_2\Phi\downarrow)$. Moreover, we have that $\mathsf{Flawed}(M\Phi\downarrow) = \mathsf{Flawed}(M_1\Phi\downarrow) \cup \mathsf{Flawed}(M_2\Phi\downarrow)$. Since $\mu(M_1) < \mu(M)$, $\mu(M_2) < \mu(M)$ and $t \in \mathsf{Flawed}(M_1\Phi\downarrow) \cup \mathsf{Flawed}(M_2\Phi\downarrow)$, we conclude by applying our inductive hypothesis on M_1 (or M_2).

Case $\mathsf{f} \in \{\mathsf{pk},\mathsf{vk}\}$: In this case, $M\Phi\downarrow = \mathsf{f}(M_1\Phi\downarrow)$ and we have that $\mathsf{Flawed}(M\Phi\downarrow) = \emptyset$. Hence the result trivially holds.

Case $\mathsf{f} \in \{\mathsf{senc},\mathsf{aenc},\mathsf{sign}\}$: In such a case, we have that $M\Phi\downarrow = \mathsf{f}(M_1\Phi\downarrow, M_2\Phi\downarrow)$. We need to distinguish whether $\mathsf{root}(M_1\Phi\downarrow) = \mathsf{tag}_i$ for some $i \in \{1,\ldots,p\}$ or not.

If $\mathsf{root}(M_1\Phi\downarrow) = \mathsf{tag}_i$ for some $i \in \{1,\ldots,p\}$, then there exists u_1 such that $M_1\Phi\downarrow = \mathsf{tag}_i(u_1)$. Hence, we have that $\mathsf{Flawed}(M_1\Phi\downarrow) = \mathsf{Flawed}(u_1)$. We have also that:

$$\mathsf{Flawed}(M\Phi\downarrow) = \mathsf{Flawed}(u_1) \cup \mathsf{Flawed}(M_2\Phi\downarrow).$$

We deduce that $t \in \mathsf{Flawed}(M_1\Phi\downarrow)$ or $t \in \mathsf{Flawed}(M_2\Phi\downarrow)$. Since $\mu(M_1) < \mu(M)$ and $\mu(M_2) < \mu(M)$, we conclude by applying our inductive hypothesis on M_1 or M_2.

Otherwise $\mathsf{root}(M_1\Phi\downarrow) \notin \{\mathsf{tag}_1, \ldots, \mathsf{tag}_p\}$. In such a case, $\mathsf{Flawed}(M\Phi\downarrow) = \mathsf{Flawed}(M_1\Phi\downarrow) \cup \mathsf{Flawed}(M_2\Phi\downarrow) \cup \{M\Phi\downarrow\}$. If $t = M\Phi\downarrow$, we have that $t_1 = M_1\Phi\downarrow$, $t_2 = M_2\Phi\downarrow$ and $\mu(M_1) < \mu(M)$, $\mu(M_2) < \mu(M)$. Thus the result holds. If $t \in \mathsf{Flawed}(M_1\Phi\downarrow) \cup \mathsf{Flawed}(M_2\Phi\downarrow)$, we conclude by applying our inductive hypothesis on M_1 or M_2.

Case $\mathsf{f} = \mathsf{h}$*:* This case is analogous to the previous one and can be handled similarly.

Case $\mathsf{f} \in \{\mathsf{sdec}, \mathsf{adec}, \mathsf{check}\}$*:* In such a case, we have to distinguish two cases depending on whether f is reduced in $M\Phi\downarrow$, or not.

If f is not reduced, *i.e.* $M\Phi\downarrow = \mathsf{f}(M_1\Phi\downarrow, M_2\Phi\downarrow)$, then we have that

$$\mathsf{Flawed}(M\Phi\downarrow) = \{M\Phi\downarrow\} \cup \mathsf{Flawed}(M_1\Phi\downarrow) \cup \mathsf{Flawed}(M_2\Phi\downarrow).$$

Thus if $t = M\Phi\downarrow$, we have that $t_1 = M_1\Phi\downarrow$, $t_2 = M_2\Phi\downarrow$ and $\mu(M_1) < \mu(M)$, $\mu(M_2) < \mu(M)$. Thus the result holds. Otherwise, we have that $t \in \mathsf{Flawed}(M_1\Phi\downarrow)$ or $t \in \mathsf{Flawed}(M_2\Phi\downarrow)$. Since $\mu(M_1) < \mu(M)$, $\mu(M_2) < \mu(M)$, we can conclude by applying our inductive hypothesis on M_1 or M_2.

If f is reduced, then we have that $M_1\Phi\downarrow = \mathsf{f}'(u_1, u_2)$ with $M\Phi\downarrow = u_1$ and $\mathsf{f}' \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\}$. If $\mathsf{root}(u_1) = \mathsf{tag}_i$ for some $i \in \{1, \ldots, p\}$, then we have that there exists u'_1 such that $u_1 = \mathsf{tag}_i(u'_1)$, $\mathsf{Flawed}(M\Phi\downarrow) = \mathsf{Flawed}(u'_1)$ and $\mathsf{Flawed}(M_1\Phi\downarrow) = \mathsf{Flawed}(u'_1) \cup \mathsf{Flawed}(u_2)$. Thus, we have that $\mathsf{Flawed}(M\Phi\downarrow) \subseteq \mathsf{Flawed}(M_1\Phi\downarrow)$. Otherwise, if $\mathsf{root}(u_1) \notin \{\mathsf{tag}_1, \ldots, \mathsf{tag}_p\}$, then we have that

$$\mathsf{Flawed}(M_1\Phi\downarrow) = \{M_1\Phi\downarrow\} \cup \mathsf{Flawed}(u_1) \cup \mathsf{Flawed}(u_2)$$

and $\mathsf{Flawed}(M\Phi\downarrow) = \mathsf{Flawed}(u_1)$. Thus, $\mathsf{Flawed}(M\Phi\downarrow) \subseteq \mathsf{Flawed}(M_1\Phi\downarrow)$. In both cases, we have that $\mathsf{Flawed}(M\Phi\downarrow) \subseteq \mathsf{Flawed}(M_1\Phi\downarrow)$ and since $\mu(M_1) < \mu(M)$, we can conclude by applying our inductive hypothesis on M_1.

In the following lemma, we will use the factors of the signature only composed of $\langle\,\rangle$, denoted $Fct_{\langle\,\rangle}$. Typically, for all terms u, for all context built only on $\langle\,\rangle$, for all terms $u_1, \ldots, u_n$, if $u = C[u_1, \ldots, u_n]$ and for all $k \in \{1, \ldots, n\}$, $\mathsf{root}(u_i) \neq \langle\,\rangle$ then $Fct_{\langle\,\rangle}(u) = \{u_1, \ldots, u_n\}$.

Lemma 14. *Let $(\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ be a derived well-tagged extended process w.r.t $\prec$ and col. Assume that for all assignment variables z, $\mathit{new}\mathcal{E}.\Phi \not\vdash z\sigma\downarrow$. Let M such that $fv(M) \subseteq \mathrm{dom}(\Phi)$, $fn(M) \cap \mathcal{E} = \emptyset$. For all $\{\gamma, \gamma'\} = \{\alpha, \beta\}$, for all $t \in Fct_\gamma(M\Phi\downarrow)$, if $t \notin Fct_{\langle\,\rangle}(M\Phi\downarrow)$ and for all assignment variable z, for all $w \in \mathrm{dom}(\Phi)$, $z \prec w$ and $\mu(w) \leq \mu(M)$ implies $z\sigma\downarrow \neq t$ then there exists M' such that $\mu(M') < \mu(M)$, $fn(M) \cap \mathcal{E} = \emptyset$ and $t \in Fct_{\langle\,\rangle}(M'\Phi\downarrow)$.*

Proof. We do a proof by induction on $\mu(M)$:

Base case $\mu(M) = (0, 1)$*:* In this case, we have that $M \in \mathcal{N}$ which means that $M\Phi\downarrow = M \in \mathcal{N}$ and $Fct_\gamma(M\Phi\downarrow) = \emptyset$. Thus the result holds.

Base case $\mu(M) = (j, 1)$*:* In this case, we have $M = w_j$. Let $\{\gamma, \gamma'\} = \{\alpha, \beta\}$. Let $t \in Fct_\gamma(M\Phi\downarrow)$ such that $t \notin Fct_{\langle\,\rangle}(M\Phi\downarrow)$. We do a case analysis on $col(w_j)$:

Case $col(w_j) \in \gamma$: In this case, since for all assignment variable z, for all $w \in \mathrm{dom}(\Phi)$, $z \prec w$ and $\mu(w) \leq \mu(M)$ implies $z\sigma{\downarrow} \neq t$, than we can deduce that for all assignment variables $z \prec w_j$, $z\sigma{\downarrow} \neq t$. Thus by Lemma 12, we obtain that there exists M' such that $fv(M') \subseteq \mathrm{dom}(\Phi) \cap \{z \mid z \prec x\}$, $fn(M') \cap \mathcal{E} = \emptyset$ and $t \in Fct_\gamma(M'\Phi{\downarrow})$. $fv(M') \subseteq \mathrm{dom}(\Phi) \cap \{z \mid z \prec x\}$ implies that $\mu(M') = (k, k')$ with $k < j$ and so $\mu(M') < \mu(M)$. If $t \in Fct_{\langle\ \rangle}(M'\Phi{\downarrow})$ then the result holds. Otherwise, we can apply our inductive hypothesis on t and M' and so the result holds.

Case $col(w_j) \in \gamma'$: Since $t \notin Fct_{\langle\ \rangle}(M\Phi{\downarrow})$, we deduce that there exists $u \in Fct_{\langle\ \rangle}(M\Phi{\downarrow})$ s.t. $\mathsf{tagroot}(u) = \gamma$ and $t \in Fct_\gamma(u)$. Note that $\mathsf{tagroot}(u) \notin \gamma' \cup \{0\}$ otherwise it would contradict the fact that $t \in Fct_\gamma(M\Phi{\downarrow})$. But $u \in Fct_{\gamma'}(M\Phi{\downarrow})$. Moreover, $u \in Fct_{\langle\ \rangle}(M\Phi{\downarrow})$ implies that u is deducible in $\mathtt{new}\ \mathcal{E}.\Phi$. Thus we deduce that for all assignment variables z, $z\sigma{\downarrow} \neq u$. By applying the same proof as case $col(w_j) \in \gamma$, we deduce that there exists M' such that $fn(M') \cap \mathcal{E} = \emptyset$, $\mu(M') < \mu(M)$ and $u \in Fct_{\langle\ \rangle}(M'\Phi{\downarrow})$. But $t \in Fct_\gamma(u)$, $\mathsf{tagroot}(u) = \gamma$ and $u \in Fct_{\langle\ \rangle}(M'\Phi{\downarrow})$ implies that $t \in Fct_\gamma(M'\Phi{\downarrow})$ and $t \notin Fct_{\langle\ \rangle}(M'\Phi{\downarrow})$. Hence we can apply our inductive hypothesis on M' and t which allows us to conclude.

Inductive step $\mu(M) > (j, 1)$: In such a case, we have that $M = \mathsf{f}(M_1, \ldots, M_n)$. Let $t \in Fct_\gamma(M\Phi{\downarrow})$ such that $t \notin Fct_{\langle\ \rangle}(M\Phi{\downarrow})$. We do a case analysis on f.

Case $\mathsf{f} \in \Sigma_i \cup \Sigma_{\mathsf{tag}_i}$ for some $i \in \gamma$: In such a case, $M\Phi{\downarrow} = \mathsf{f}(M_1\Phi{\downarrow}, \ldots, M_n\Phi{\downarrow}){\downarrow}$. By definition, we know that for all $t \in Fct_\gamma(M\Phi{\downarrow})$, we have that $\mathsf{root}(t) \notin \Sigma_i \cup \Sigma_{\mathsf{tag}_i}$. Thus, thanks to Lemma 10, we deduce that there exists

$$Fct_\gamma(M\Phi{\downarrow}) \subseteq Fct_\gamma(M_1\Phi{\downarrow}) \cup \ldots \cup Fct_\gamma(M_n\Phi{\downarrow}).$$

Thus there exists $k \in \{1, \ldots, n\}$ such that $t \in Fct_\gamma(M_k\Phi{\downarrow})$. If $t \in Fct_{\langle\ \rangle}(M_k\Phi{\downarrow})$ then the result holds, else we apply our inductive hypothesis on t and M_k and so the result also holds.

Case $\mathsf{f} \in \Sigma_i \cup \Sigma_{\mathsf{tag}_i}$ for some $i \notin \gamma$: In such a case, $M\Phi{\downarrow} = \mathsf{f}(M_1\Phi{\downarrow}, \ldots, M_n\Phi{\downarrow}){\downarrow}$. We assumed that $t \notin Fct_{\langle\ \rangle}(M\Phi{\downarrow})$ hence there exists $u \in Fct_{\langle\ \rangle}(M\Phi{\downarrow})$ s.t. $\mathsf{tagroot}(u) = \gamma$ and $t \in Fct_\gamma(u)$. But it also implies that $\mathsf{tagroot}(M\Phi{\downarrow}) \in \gamma \cup \{0\}$. Hence, by applying Lemma 2, we deduce that there exists $k \in \{1, \ldots, n\}$ such that $M\Phi{\downarrow} \in st(M_k\Phi{\downarrow})$. Moreover, it also implies that $u \in Fct'_\gamma(M_k\Phi{\downarrow})$.

If $u \in Fct_{\langle\ \rangle}(M_k\Phi{\downarrow})$ then we deduce that $\mathsf{root}(M_k\Phi{\downarrow}) \notin \gamma'$ and so, by Lemma 2, $M_k\Phi{\downarrow} = M\Phi{\downarrow}$. Since we had $t \notin Fct_{\langle\ \rangle}(M\Phi{\downarrow})$, then we also have $t \notin Fct_{\langle\ \rangle}(M_k\Phi{\downarrow})$ and so we conclude by applying our inductive hypothesis on t and M_k.

if $u \notin Fct_{\langle\ \rangle}(M_k\Phi{\downarrow})$ then we can apply our inductive hypothesis on u, γ' and M_k. Indeed, since $u \in Fct_{\langle\ \rangle}(M\Phi{\downarrow})$, then u is deducible in $\mathtt{new}\ \mathcal{E}.\Phi$ and so we deduce that for all assignment variable z, $z\sigma{\downarrow} \neq u$. Hence we obtain that there exists M' such that $\mu(M') < \mu(M_k)$, $fn(M) \cap \mathcal{E} = \emptyset$ and $u \in Fct_{\langle\ \rangle}(M'\Phi{\downarrow})$. But $t \in Fct_\gamma(u)$ and $u \in Fct_{\langle\ \rangle}(M'\Phi{\downarrow})$. Hence we deduce that $t \in Fct_\gamma(M'\Phi{\downarrow})$ and $t \notin Fct_{\langle\ \rangle}(M'\Phi{\downarrow})$. We conclude by applying once again our inductive hypothesis but on t, γ and M'.

Case $\mathsf{f} = \langle\ \rangle$: In such a case, $M\Phi{\downarrow} = \mathsf{f}(M_1\Phi{\downarrow}, M_2\Phi{\downarrow})$. Moreover, we have that $Fct_\gamma(M\Phi{\downarrow}) = Fct_\gamma(M_1\Phi{\downarrow}) \cup Fct_\gamma(M_2\Phi{\downarrow})$. Since $\mu(M_1) < \mu(M)$, $\mu(M_2) < \mu(M)$

and $t \in Fct_\gamma(M_1\Phi\downarrow) \cup Fct_\gamma(M_2\Phi\downarrow)$, we conclude by applying our inductive hypothesis on t and M_1 (or M_2).

Case $\mathsf{f} \in \{\mathsf{pk}, \mathsf{vk}\}$*:* In this case, $M\Phi\downarrow = \mathsf{f}(M_1\Phi\downarrow)$ and we have that $Fct_\gamma M\Phi\downarrow = \emptyset$. Hence the result trivially holds.

Case $\mathsf{f} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\}$*:* In such a case, we have that $M\Phi\downarrow = \mathsf{f}(M_1\Phi\downarrow, M_2\Phi\downarrow)$. We need to distinguish whether $\mathsf{root}(M_1\Phi\downarrow) = \mathsf{tag}_i$ for some $i \in \{1, \ldots, p\}$ or not.

If $\mathsf{root}(M_1\Phi\downarrow) = \mathsf{tag}_i$ for some $i \in \{1, \ldots, p\}$, then there exists u_1 such that $M_1\Phi\downarrow = \mathsf{tag}_i(u_1)$. Assume first that $i \in \gamma'$. In such a case $Fct_\gamma(M\Phi\downarrow) = \{Fct_\gamma(M\Phi\downarrow)\}$ and $Fct_{\langle\ \rangle}(M\Phi\downarrow) = \{Fct_{\langle\ \rangle}(M\Phi\downarrow)\}$. Hence it contradicts the fact that $t \notin Fct_{\langle\ \rangle}(M\Phi\downarrow)$. We can thus deduce that $i \in \gamma$. But in such a case, we have that $Fct_\gamma(M_1\Phi\downarrow) = Fct_\gamma(u_1)$ and:

$$Fct_\gamma(M\Phi\downarrow) = Fct_\gamma(u_1) \cup Fct_\gamma(M_2\Phi\downarrow).$$

We deduce that $t \in Fct_\gamma(M_1\Phi\downarrow)$ or $t \in Fct_\gamma(M_2\Phi\downarrow)$. Since $\mu(M_1) < \mu(M)$ and $\mu(M_2) < \mu(M)$, we conclude by applying our inductive hypothesis on M_1 or M_2.

Otherwise $\mathsf{root}(M_1\Phi\downarrow) \notin \{\mathsf{tag}_1, \ldots, \mathsf{tag}_p\}$. In such a case, $Fct_\gamma(M\Phi\downarrow) = \{M\Phi\downarrow\}$ and $Fct_{\langle\ \rangle}(M\Phi\downarrow) = \{M\Phi\downarrow\}$. But we assume that $t \notin Fct_{\langle\ \rangle}(M\Phi\downarrow)$ hence this case is impossible.

Case $\mathsf{f} = \mathsf{h}$*:* This case is analogous to the previous one and can be handled similarly.

Case $\mathsf{f} \in \{\mathsf{sdec}, \mathsf{adec}, \mathsf{check}\}$*:* In such a case, we have to distinguish two cases depending on whether f is reduced in $M\Phi\downarrow$, or not.

If f is not reduced, *i.e.* $M\Phi\downarrow = \mathsf{f}(M_1\Phi\downarrow, M_2\Phi\downarrow)$, then we have that

$$Fct_\gamma(M\Phi\downarrow) = \{M\Phi\downarrow\}.$$

Once again this is in contradiction with our hypothesis that $t \notin Fct_{\langle\ \rangle}(M\Phi\downarrow)$.

We now focus on the case where f is reduced: we have that $M_1\Phi\downarrow = \mathsf{f}'(u_1, u_2)$ with $M\Phi\downarrow = u_1$ and $\mathsf{f}' \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\}$. We have to do a case analysis on $\mathsf{root}(u_1)$:

- if $\mathsf{root}(u_1) = \mathsf{tag}_i$ for some $i \in \gamma$. In such a case, there exists u_1' such that $u_1 = \mathsf{tag}_i(u_1')$, $Fct_\gamma(M\Phi\downarrow) = Fct_\gamma(u_1')$ and $Fct_\gamma(M_1\Phi\downarrow) = Fct_\gamma(u_1') \cup Fct_\gamma(u_2)$. Thus we deduce that $Fct_\gamma(M\Phi\downarrow) \subseteq Fct_\gamma(M_1\Phi\downarrow)$. We can conclude thanks to our inductive hypothesis on t and M_1.
- if $\mathsf{root}(u_1) = \mathsf{tag}_i$ for some $i \notin \gamma$. In such a case, $Fct_\gamma(M\Phi\downarrow) = \{M\Phi\downarrow)$ which contradicts the hypothesis $t \notin Fct_{\langle\ \rangle}(M\Phi\downarrow)$.
- otherwise, $\mathsf{root}(u_1) \notin \{\mathsf{tag}_1, \ldots, \mathsf{tag}_p\}$, then we have that $\mathsf{f}'(u_1, u_2) \in \mathsf{Flawed}(M_1\Phi\downarrow)$. By Lemma 13, we deduce that there exists M' such that $\mu(M') < \mu(M_1)$, $fn(M') \cap \mathcal{E} = \emptyset$ and $M'\Phi\downarrow = u_1$. Since $u_1 = M\Phi\downarrow$ and $\mu(M') < \mu(M)$ then we can apply our inductive hypothesis on t, α and M' and so the result holds.

Lemma 15. *Let $A = (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ be a derived well-tagged process, and let $(\rho_\alpha, \rho_\beta)$ be compatible with A. Let u be a ground term in normal form that do not use* *names in $\mathcal{E}_\alpha \uplus \mathcal{E}_\beta$. We have that there exists a context C (possibly a hole) built only using $\langle\ \rangle$, and terms $u_1, \dots, u_m$ such that $u = C[u_1, \dots, u_m]$, and for all $i \in \{1, \dots, m\}$,*

- *either $u_i \in \mathsf{Flawed}(u)$;*
- *or $u_i \in Fct_{\Sigma_0}(u)$ and $\delta_\alpha(u_i) = \delta_\beta(u_i)$,*
- *or $u_i = \mathsf{f}(n)$ for some $\mathsf{f} \in \{\mathsf{pk}, \mathsf{vk}\}$ and $n \in \mathcal{N}$,*
- *or $u_i \in \mathrm{dom}(\rho^+_\alpha) \cup \mathrm{dom}(\rho^+_\beta)$.*

Proof. Let u a ground term in normal form and let $\{v_1, \dots, v_n\} = Fct_{\Sigma_0}(u)$. Thus there exists a context D (possibly a hole) built on Σ_0 such that $u = D[v_1, \dots, v_n]$. We now prove the result by induction on $|D|$.

Base case $|D| = 0$: We show that the result holds and in such a case the context C is reduced to a hole. Since $|D| = 0$, we know that $Fct_{\Sigma_0}(u) = u$ and so either $\mathsf{tagroot}(u) = i$ with $i \in \{1, \dots, p\}$ or $\mathsf{tagroot}(u) = \perp$. If $u \in \mathrm{dom}(\rho^+_\alpha) \cup \mathrm{dom}(\rho^+_\beta)$, then the result trivially holds. Otherwise, we have that $\delta_\alpha(u) = \delta_\beta(u)$ by definition of δ_α and δ_β. Hence the result holds.

Inductive step $|D| > 0$: There exists $\mathsf{f} \in \Sigma_0$, and $v_1, \dots, v_k$ such that $u = \mathsf{f}(u_1, \dots, u_k)$. We do a case analysis on f.

Case $\mathsf{f} = \langle\ \rangle$: In such a case, there exist two contexts D_1, D_2 (possibly holes) built on Σ_0 such that:

- $D = \langle D_1, D_2\rangle$ with $|D_1|, |D_2| < |D|$,
- $u_1 = D_1[v^1_1, \dots, v^1_{n_1}]$ and $\{v^1_1, \dots, v^1_{n_1}\} = Fct_{\Sigma_0}(u_1)$,
- $u_2 = D_1[v^2_1, \dots, v^2_{n_1}]$ and $\{v^2_1, \dots, v^2_{n_2}\} = Fct_{\Sigma_0}(u_2)$

By applying our inductive hypothesis on u_1 and u_2, we know that there exist two contexts C_1 and C_2. Since

- $\mathsf{Flawed}(u) = \mathsf{Flawed}(u_1) \cup \mathsf{Flawed}(u_2)$, and
- $Fct_{\Sigma_0}(u) = Fct_{\Sigma_0}(u_1) \uplus Fct_{\Sigma_0}(u_2)$,

we conclude that $C = \langle C_1, C_2\rangle$ satisfies all the conditions stated in the lemma.

Case $\mathsf{f} \in \{\mathsf{pk}, \mathsf{vk}\}$ and $u = \mathsf{f}(n)$ for some $n \in N$: The result trivially hold by choosing the context C to be a hole.

Otherwise, we have that

$$\mathsf{Flawed}(u) = \{u\} \cup \mathsf{Flawed}(u_1) \cup \ldots \cup \mathsf{Flawed}(u_k).$$

Since $u \in \mathsf{Flawed}(u)$, we can choose C to be the context reduced to a hole. The result trivially holds.

Lemma 16. *Let $A = (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ be a derived well-tagged extended process, and let $(\rho_\alpha, \rho_\beta)$ be compatible with A. Let M be a term such that $fv(M) \subseteq \mathrm{dom}(\Phi)$ and $fn(M) \cap \mathcal{E} = \emptyset$. We assume that $\mathcal{E} = \mathcal{E}_0 \uplus \mathcal{E}_\alpha \uplus \mathcal{E}_\beta$, $fn(\Phi) \cap (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta) = \emptyset$, and one of the two following conditions is satisfied:*

1. $\mathtt{new}\ \mathcal{E}.\Phi \not\vdash k$ *for any* $k \in K_S$*; or*
2. $\mathtt{new}\ \mathcal{E}.\delta(\Phi\!\downarrow) \not\vdash k$ *for any* $k \in \delta_\alpha(K_S) \cup \delta_\beta(K_S)$*.*

with $K_S = \{t, \mathsf{pk}(t), \mathsf{vk}(t) \mid t \text{ ground}, t \in \mathrm{dom}(\rho^+_\alpha) \cup \mathrm{dom}(\rho^+_\beta)\}$*. We have that* $\delta_\gamma(M\Phi\!\downarrow) = M\delta(\Phi\!\downarrow)\!\downarrow$ *with* $\gamma \in \{\alpha, \beta\}$*.*

Proof. Let $\Phi\!\downarrow = \{w_1 \rhd u_1, \ldots, w_n \rhd u_n\}$. We prove this result by induction on $\mu(M)$:

Base case $\mu(M) = (0, 0)$*:* There exists no term M such that $|M| = 0$, thus the result holds.

Inductive step $\mu(M) > (0, 0)$*:* We first prove there exists $\gamma \in \{\alpha, \beta\}$ such that $\delta_\gamma(M\Phi\!\downarrow) = M\delta(\Phi\!\downarrow)\!\downarrow$ and then we show that $\delta_\alpha(M\Phi\!\downarrow) = \delta_\beta(M\Phi\!\downarrow)$.

Assume first that $|M| = 1$, *i.e.* either $M \in \mathcal{N}$ or there exists $j \in \{1, \ldots, n\}$ such that $M = w_j$.
Case $M \in \mathcal{N}$. In such a case, we have that $M\Phi\!\downarrow = M$, and $M \notin \mathcal{E}$. Hence, we have that $\mathtt{new}\ \mathcal{E}.\Phi \vdash M$ and also that $\mathtt{new}\ \mathcal{E}.\delta(\Phi\!\downarrow) \vdash M$. In case condition 1 is satisfied, we easily deduce that $M \notin K_S$. Otherwise, we know that the condition 2 is satisfied, and thus $M \notin \delta_\alpha(K_S) \cup \delta_\beta(K_S)$. Again, we want to conclude that $M \notin K_S$. Assume that this is not the case, *i.e.* $M \in K_S$. This means that M is a name in $\mathrm{dom}(\rho^+_\alpha)$ (or $\mathrm{dom}(\rho^+_\beta)$). Hence, we have that $\delta_\beta(M) \in \delta_\beta(K_S)$, and $\delta_\beta(M) = M$. Hence, we deduce that $M \in \delta_\beta(K_S)$, and this leads to a contradiction, since in such a case, by hypothesis M can not be deducible from $\mathtt{new}\ \mathcal{E}.\delta(\Phi\!\downarrow)$. Thus, in any case, we have that $M \notin K_S$, and thus $M \notin \mathrm{dom}(\rho^+_\alpha) \cup \mathrm{dom}(\rho^+_\beta)$. Hence, we have that $\delta_\gamma(M\Phi\!\downarrow) = \delta_\gamma(M) = M = M\delta(\Phi\!\downarrow)\!\downarrow$ for any $\gamma \in \{\alpha, \beta\}$.
Case $M = w_j$ *for some* $j \in \{1, \ldots, n\}$. We know that w_j is colored with $\gamma \in \{\alpha, \beta\}$. Hence, we have that $w_j\delta(\Phi\!\downarrow) = \delta_\gamma(w_j\Phi\!\downarrow)$. Since u_j is in normal form, then by Lemma 6, we know that $\delta_\gamma(w_j\Phi)$ is also in normal form. Thus, we have that $\delta_\gamma(M\Phi\!\downarrow) = M\delta(\Phi\!\downarrow)\!\downarrow$.

Otherwise, if $|M| > 1$, then there exists a symbol f and $M_1, \ldots, M_n$ such that $M = \mathsf{f}(M_1, \ldots, M_n)$. We do a case analysis on f.

Case $\mathsf{f} \in \Sigma_i \cup \Sigma_{\mathsf{tag}_i}$ *with* $i \in \{1, \ldots, p\}$. Consider $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. In such a case, let $t = \mathsf{f}(M_1\Phi\!\downarrow, \ldots, M_n\Phi\!\downarrow)$. Since $\mathsf{f} \in \Sigma_i$ (resp. Σ_{tag_i}), then there exists a context C built upon Σ_i (resp. Σ_{tag_i}) such that $t = C[u_1, \ldots, u_m]$ and $u_1, \ldots, u_m$ are factor of t in normal form. By Lemma 2, we know that there exists a context D (possibly a hole) over Σ_i (resp. Σ_{tag_i}) such that $t\!\downarrow = D[u_{i_1}, \ldots, u_{i_k}]$ with $i_1, \ldots, i_k \in \{0, \ldots, m\}$ and $u_0 = n_{min}$. But thanks to Lemma 3, 4 and 6, we also that $C[\delta_\gamma(u_1), \ldots, \delta_\gamma(u_m)]\!\downarrow = D[\delta_\gamma(u_{i_1}), \ldots, \delta_\gamma(u_{i_k})]$. But C and D are both built on Σ_i (resp. Σ_{tag_i}), thus by definition of δ_γ, we have that $\delta_\gamma(t)\!\downarrow = C[\delta_\gamma(u_1), \ldots, \delta_\gamma(u_m)]\!\downarrow$ and $\delta_\gamma(t\!\downarrow) = D[\delta_\gamma(u_{i_1}), \ldots, \delta_\gamma(u_{i_k})]$. Hence, the equality, $\delta_\gamma(t\!\downarrow) = \delta_\gamma(t)\!\downarrow$, holds. But $t\!\downarrow = M\Phi\!\downarrow$ which means that $\delta_\gamma(M\Phi\!\downarrow) = \delta_\gamma(t)\!\downarrow$. We have that:

$$\begin{aligned}\delta_\gamma(t)\!\downarrow &= \delta_\gamma(\mathsf{f}(M_1\Phi\!\downarrow, \ldots, M_n\Phi\!\downarrow))\!\downarrow \\ &= \mathsf{f}(\delta_\gamma(M_1\Phi\!\downarrow), \ldots, \delta_\gamma(M_n\Phi\!\downarrow))\!\downarrow\end{aligned}$$

Since $\mu(M_1) < \mu(M)$, $\ldots$, $\mu(M_n) < \mu(M)$, we can apply our inductive hypothesis on $M_1, \ldots, M_n$. This gives us $\delta_\gamma(t)\downarrow = \mathsf{f}(M_1\delta(\Phi\downarrow)\downarrow, \ldots, M_n\delta(\Phi\downarrow)\downarrow)\downarrow = \mathsf{f}(M_1, \ldots, M_n)\delta(\Phi\downarrow)\downarrow$. Thus we can conclude that $\delta_\gamma(M\Phi\downarrow) = \delta_\gamma(t)\downarrow = M\delta(\Phi\downarrow)\downarrow$.

Case $\mathsf{f} \in \Sigma_0 \smallsetminus \{\mathsf{sdec}, \mathsf{adec}, \mathsf{check}\}$*:* In this case, we have that $M\Phi\downarrow = \mathsf{f}(M_1\Phi\downarrow, \ldots, M_n\Phi\downarrow)$. By applying our inductive hypothesis on $M_1, \ldots, M_n$, we have that

$$\delta_\alpha(M_k\Phi\downarrow) = \delta_\beta(M_k\Phi\downarrow), \text{ for all } k \in \{1, \ldots, n\}.$$

Thus we have that $\delta_\gamma(M\Phi\downarrow) = \mathsf{f}(\delta_{\gamma'}(M_1\Phi\downarrow), \ldots, \delta_{\gamma'}(M_n\Phi\downarrow))$ with $\gamma, \gamma' \in \{\alpha, \beta\}$. Applying our inductive hypothesis on $M_1, \ldots, M_n$, we deduce that

$$\delta_\gamma(M\Phi\downarrow) = \mathsf{f}(M_1\delta(\Phi\downarrow)\downarrow, \ldots, M_n\delta(\Phi\downarrow)\downarrow) = M\delta(\Phi\downarrow)\downarrow.$$

Case $\mathsf{f} \in \{\mathsf{sdec}, \mathsf{adec}, \mathsf{check}\}$*:* If we first assume that the root occurence f is not reduced in $M\Phi\downarrow$ then the proof is similar to the previous case. Thus, we focus on the case where the root occurence of f is reduced, and we consider the case where $\mathsf{f} = \mathsf{sdec}$. The other cases can be done in a similar way. In such a situation, we know that there exist v_1, v_2 such that $M_1\Phi\downarrow = \mathsf{senc}(v_1, v_2)$, $M_2\Phi\downarrow = v_2$ and $M\Phi\downarrow = v_1$. According to the definition of δ_γ, we know that there exists $\gamma \in \{\alpha, \beta\}$ such that $\delta_\gamma(\mathsf{senc}(v_1, v_2)) = \mathsf{senc}(\delta_\gamma(v_1), \delta_\gamma(v_2))$. For such γ, we have that $\mathsf{sdec}(\delta_\gamma(M_1\Phi\downarrow), \delta_\gamma(M_2\Phi\downarrow))\downarrow = \delta_\gamma(M\Phi\downarrow)$. But by applying our inductive hypothesis on M_1 and M_2, we obtain $\delta_\gamma(M\Phi\downarrow) = \mathsf{sdec}(M_1\delta(\Phi\downarrow)\downarrow, M_2\delta(\Phi\downarrow)\downarrow)\downarrow = M\delta(\Phi\downarrow)\downarrow$.

It remains to prove that $\delta_\alpha(M\Phi\downarrow) = \delta_\beta(M\Phi\downarrow)$. We have shown that there exists $\gamma_0 \in \{\alpha, \beta\}$ such that $\delta_{\gamma_0}(M\Phi\downarrow) = M\delta(\Phi\downarrow)\downarrow$. Thanks to Lemma 15, we know that there exists a context C built over $\{\langle\rangle\}$, and $v_1, \ldots, v_m$ terms such that $M\Phi\downarrow = C[v_1, \ldots, v_m]$ and for all $i \in \{1, \ldots, m\}$:

- either $v_i \in \mathsf{Flawed}(M\Phi\downarrow)$
- or $v_i \in Fct_{\Sigma_0}(M\Phi\downarrow)$ and $\delta_\alpha(v_i) = \delta_\beta(v_i)$.
- or $v_i = \mathsf{f}(n)$ for some $\mathsf{f} \in \{\mathsf{pk}, \mathsf{vk}\}$ and $n \in \mathcal{N}$,
- or $v_i \in \mathrm{dom}(\rho^+_\alpha) \cup \mathrm{dom}(\rho^+_\beta)$.

Note that C being built upon $\{\langle\rangle\}$ means that v_i is deducible in $\mathtt{new}\ \mathcal{E}.\Phi$ for all $i \in \{1, \ldots, m\}$. Furthermore, since $C[v_1, \ldots, v_m]$ is in normal form,

$$\delta_{\gamma_0}(M\Phi\downarrow) = C[\delta_{\gamma_0}(v_1), \ldots, \delta_{\gamma_0}(v_m)].$$

But we have shown that $\delta_{\gamma_0}(M\Phi\downarrow) = M\delta(\Phi\downarrow)\downarrow$, thus $\delta_{\gamma_0}(v_i)$ is deducible from $\delta(\Phi\downarrow)$, for all $i \in \{1, \ldots, m\}$. Now, we distinguish several cases depending on which condition is fullfilled by v_i.

Case $v_i \in \mathsf{Flawed}(M\Phi\downarrow)$*:* There exists $w_1, \ldots, w_\ell$ terms and a function symbol f such that $v_i = \mathsf{f}(w_1, \ldots, w_\ell)$. By Lemma 13, there exists $N_1, \ldots, N_\ell$ such that for all $k \in \{1, \ldots, \ell\}$, $\mu(N_k) < \mu(M)$ and $N_k\Phi\downarrow = w_k$. Hence, by applying inductive hypothesis on $N_1, \ldots, N_\ell$, we obtain that $\delta_\alpha(N_k\Phi\downarrow) = \delta_\beta(N_k\Phi\downarrow)$, for all $k \in \{1, \ldots, \ell\}$. Thus, thanks to v_i being in normal form, we can conclude that $\delta_\alpha(v_i) = \delta_\beta(v_i)$.

Case $v_i \in Fct_{\Sigma_0}(M\Phi\downarrow)$: In such a case, we have that $\delta_\alpha(v_i) = \delta_\beta(v_i)$. Hence, we easily conclude.

Case $v_i = \mathsf{f}(n)$ *for some* $\mathsf{f} \in \{\mathsf{pk}, \mathsf{vk}\}$ *and* $n \in \mathcal{N}$: By hypothesis, we know that either $\mathtt{new}\, \mathcal{E}.\Phi \not\vdash k$, for all $k \in K_S$; or $\mathtt{new}\, \mathcal{E}.\delta(\Phi\downarrow) \not\vdash k$, for all $k \in \delta_\alpha(K_S) \cup \delta_\beta(K_S)$. Since we have shown that v_i is deducible from $\mathtt{new}\, \mathcal{E}.\Phi$ and $\delta_{\gamma_0}(v_i)$ is deducible from $\mathtt{new}\, \mathcal{E}.\delta(\Phi\downarrow)$, both hypotheses imply that $n \notin \mathrm{dom}(\rho_\alpha^+) \cup \mathrm{dom}(\rho_\beta^+)$, and so $\delta_\alpha(v_i) = \delta_\beta(v_i)$.

Case $v_i \in \mathrm{dom}(\rho_\alpha^+) \cup \mathrm{dom}(\rho_\beta^+)$: By hypothesis, we know that either $\mathtt{new}\, \mathcal{E}.\Phi \not\vdash k$, for all $k \in K_S$; or $\mathtt{new}\, \mathcal{E}.\delta(\Phi\downarrow) \not\vdash k$, for all $k \in \delta_\alpha(K_S) \cup \delta_\beta(K_S)$. Since we have shown that v_i is deducible from $\mathtt{new}\, \mathcal{E}.\Phi$ and $\delta_{\gamma_0}(v_i)$ is deducible from $\mathtt{new}\, \mathcal{E}.\delta(\Phi\downarrow)$, both hypotheses imply that $v_i \notin \mathrm{dom}(\rho_\alpha^+) \cup \mathrm{dom}(\rho_\beta^+)$ and lead us to a contradiction.

Corollary 4. *Let $A = (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ be a derived well-tagged extended process and let $(\rho_\alpha, \rho_\beta)$ be compatible with A, such that $\mathcal{E} = \mathcal{E}_0 \uplus \mathcal{E}_\alpha \uplus \mathcal{E}_\beta$, and $fn(\Phi) \cap (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta) = \emptyset$. The two following conditions are equivalent:*

1. *$\mathtt{new}\, \mathcal{E}.\Phi \not\vdash k$ for any $k \in K_S$; or*
2. *$\mathtt{new}\, \mathcal{E}.\delta(\Phi\downarrow) \not\vdash k$ for any $k \in \delta_\alpha(K_S) \cup \delta_\beta(K_S)$.*

with $K_S = \{t, \mathsf{pk}(t), \mathsf{vk}(t) \mid t \in \mathrm{dom}(\rho_\alpha^+) \cup \mathrm{dom}(\rho_\beta^+),\ t \text{ ground}\}$.

Proof. We prove the two implications separately.
(2) $\Rightarrow$ (1): Let $k \in K_S$ such that $\mathtt{new}\, \mathcal{E}.\Phi \vdash k$. In such a case, there exists M such that $fv(M) \subseteq \mathrm{dom}(\Phi)$, $fn(M) \cap \mathcal{E} = \emptyset$, and $M\Phi\downarrow = k\downarrow$. We assume w.l.o.g. that $k \in \{t, \mathsf{pk}(t), \mathsf{vk}(t) \mid t \in \mathrm{dom}(\rho_\alpha^+)$ and t ground$\}$. Let $\gamma \in \{\alpha, \beta\}$. By Lemma 4, we have that $\delta_\gamma(M\Phi\downarrow) = \delta_\gamma(k\downarrow)$. Thanks to Lemma 16, we have that $\delta_\gamma(M\Phi\downarrow) = M\delta(\Phi\downarrow)\downarrow$, and by Definition of δ_γ, we have that $\delta_\gamma(k\downarrow) \in \delta_\gamma(K_S)$. Thus, we deduce that there exists $k' \in \delta_\gamma(K_S)$ such that $\mathtt{new}\, \mathcal{E}.\delta(\Phi\downarrow) \vdash k'$.

(1) $\Rightarrow$ (2):Let $k \in \delta_\gamma(K_S)$ with $\gamma \in \{\alpha, \beta\}$, and M be a term such that $fv(M) \subseteq \mathrm{dom}(\Phi)$, $fn(M) \cap \mathcal{E} = \emptyset$, and $M\delta(\Phi\downarrow)\downarrow = k\downarrow$. $k \in \delta_\gamma(K_S)$ implies the existence of $k' \in K_S$ such that $k = \delta_\gamma(k')$, and thus such that $M\delta(\Phi\downarrow)\downarrow = \delta_\gamma(k')\downarrow$. Thanks to Lemma 16, we have that $\delta_\gamma(M\Phi\downarrow)\downarrow = \delta_\gamma(k')\downarrow$. Now, if $k' \in K_S$ there must exist $k'' \in dom(\rho_{\gamma'})$ such that either $k' = k''$, or $k' = \mathsf{pk}(k'')$, or $k' = \mathsf{vk}(k'')$. In any case, because $\rho_{\gamma'}$ is in normal form, we know that $k''\downarrow = k''$ and thus that $k'\downarrow = k'$. Hence $M\delta(\Phi\downarrow)\downarrow = \delta_\gamma(k'\downarrow)\downarrow$. But, then according to Lemma 6, $\delta_\gamma(M\Phi\downarrow) = \delta_\gamma(M\Phi\downarrow)\downarrow = \delta_\gamma(k'\downarrow)\downarrow = \delta_\gamma(k'\downarrow)$. Finally, thanks to Lemma 4 we can derive that $M\Phi\downarrow = k'\downarrow = k'$. This implies that $M\Phi\downarrow \in K_S$, and thus there is a term in K_S that is deducible from $\mathtt{new}\, \mathcal{E}.\Phi$.

Corollary 5. *Let $A = (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ be a derived extended process and let $(\rho_\alpha, \rho_\beta)$ be compatible with A such that $\mathcal{E} = \mathcal{E}_0 \uplus \mathcal{E}_\alpha \uplus \mathcal{E}_\beta$, $fn(\Phi) \cap (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta) = \emptyset$, and $\mathtt{new}\, \mathcal{E}.\Phi \not\vdash k$ for any $k \in K_S$. We have that $\mathtt{new}\, \mathcal{E}.\Phi \sim \mathtt{new}\, \mathcal{E}.\delta(\Phi\downarrow)$.*

Proof. The proof directly follows from Lemmas 4 and 16. Indeed, $M\Phi\downarrow = N\Phi\downarrow$ is equivalent to $\delta_\gamma(M\Phi\downarrow) = \delta_\gamma(N\Phi\downarrow)$ (thanks to Lemma 4), which is equivalent to $M\delta(\Phi\downarrow)\downarrow = N\delta(\Phi\downarrow)\downarrow$ (thanks to Lemma 16).

E.6 Proof of Theorem 5

The goal of this section is to prove Theorem 5. We first state and prove two propositions.

Let $S = (\mathcal{E}_S; \mathcal{P}_S; \Phi_S; \sigma_S)$ and $D = (\mathcal{E}_D; \mathcal{P}_D; \Phi_D; \sigma_D)$. We say that $D = \delta(S)$ if $\mathcal{E}_S = \mathcal{E}_D$, $\mathcal{P}_D = \delta(\mathcal{P}_S)$, $\Phi_D{\downarrow} = \delta(\Phi_S{\downarrow})$, and $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$.

Proposition 1. *Let P_0 be a plain coloured process without replication and such that $bn(P_0) = fv(P_0) = \emptyset$. Let B_0 be an extended coloured biprocess such that:*

- $S_0 = (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta \uplus \mathcal{E}_0; [\![P_0]\!]; \emptyset; \emptyset) \stackrel{\mathsf{def}}{=} \mathsf{fst}(B_0)$,
- $D_0 = (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta \uplus \mathcal{E}_0; P_0'; \emptyset; \emptyset) \stackrel{\mathsf{def}}{=} \mathsf{snd}(B_0)$*, and*
- $D_0 = \delta^{\rho_\alpha^+, \rho_\beta^+}(S_0)$ *for some* $(\rho_\alpha, \rho_\beta)$*, and*
- D_0 *does not reveal the value of its assignments w.r.t.* $(\rho_\alpha, \rho_\beta)$.

For any extended process $S = (\mathcal{E}_S; \mathcal{P}_S; \Phi_S; \sigma_S)$ such that $S_0 \stackrel{\mathsf{tr}}{\Rightarrow} S$ with $(\rho_\alpha, \rho_\beta)$ compatible with S, there exists a biprocess B and an extended process $D = (\mathcal{E}_D; \mathcal{P}_D; \Phi_D; \sigma_D)$ such that $B_0 \stackrel{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} B$, $\mathsf{fst}(B) = S$, $\mathsf{snd}(B) = D$, $D = \delta(S)$ and with $(\rho_\alpha, \rho_\beta)$ compatible with D.

Proof. Let $\mathcal{E} = \mathcal{E}_0 \uplus \mathcal{E}_\alpha \uplus \mathcal{E}_\beta$. We show the result by induction on the length of the derivation. The base case when $S = S_0$ is trivial. We simply conclude by considering $B = B_0$, and $D = D_0$. Now, we assume that $S_0 \stackrel{\mathsf{tr}'}{\Rightarrow} S'$ such that $(\rho_\alpha, \rho_\beta)$ is compatible with S'. This means that there exists S', tr and ℓ such that:

$$S_0 \stackrel{\mathsf{tr}}{\Rightarrow} S \xrightarrow{\ell} S' \text{ with } \mathsf{tr}' = \mathsf{tr} \cdot \ell$$

Moreover, we have that $(\rho_\alpha, \rho_\beta)$ is compatible with S.

By induction hypothesis, we have that there exists an extended biprocess B and an extended process D such that $\mathsf{fst}(B) = S$, $\mathsf{snd}(B) = D$, $B_0 \stackrel{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} B$, and $D = \delta(S)$. We will show by case analysis on the rule involved in $S \xrightarrow{\ell} S'$ that exists a biprocess B and an extended process $D' = (\mathcal{E}_D'; \mathcal{P}_D'; \Phi_D'; \sigma_D')$ such that $B_0 \stackrel{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} B'$, $\mathsf{fst}(B') = S'$, $\mathsf{snd}(B') = D'$, $D' = \delta(S')$. Then it will remain to prove that $(\rho_\alpha, \rho_\beta)$ compatible with D'. To do so, we rely on the fact that $\delta(\sigma_{S'}{\downarrow}) = \sigma_{D'}{\downarrow}$. In particular, by Lemma 4, we have that for all assignment variables z, z', $z\sigma_{S'}{\downarrow} = z'\sigma_{S'}{\downarrow}$ is equivalent to $\delta(z\sigma_{S'}{\downarrow}) = \delta(z'\sigma_{S'}{\downarrow})$ which is also equivalent to $z\delta(\sigma_{S'}{\downarrow}) = z'\delta(\sigma_{S'}{\downarrow})$. Moreover, since $(\rho_\alpha, \rho_\beta)$ is compatible with S', then for all assignment variable $z \in \mathrm{dom}(\rho_\gamma)$, either $\mathsf{tagroot}(z\sigma_{S'}{\downarrow}) = \perp$ or $\mathsf{tagroot}(z\sigma_{S'}{\downarrow}) \notin \gamma \cup \{0\}$. Thus, by Lemma 6, we deduce that either $\mathsf{tagroot}(\delta_\gamma'(z\sigma_{S'}{\downarrow})) = \perp$ or $\mathsf{tagroot}(\delta_\gamma'(z\sigma_{S'}{\downarrow})) \notin \gamma \cup \{0\}$. This allows us to conclude that either $\mathsf{tagroot}(z\sigma_{D'}) = \perp$ or $\mathsf{tagroot}(z\sigma_{D'}{\downarrow}) \notin \gamma \cup \{0\}$, and so that $(\rho_\alpha, \rho_\beta)$ is compatible with D'.

Let's now prove the core part of the result. Let $S = (\mathcal{E}_S; \mathcal{P}_S; \Phi_S; \sigma_S)$ and $S' = (\mathcal{E}_S'; \mathcal{P}_S'; \Phi_S; \sigma_S')$.

Case of the rule OUT-T. In such a case, we have that $\mathcal{E}_S' = \mathcal{E}_S = \mathcal{E}$, $\sigma_S' = \sigma_S$, $\mathcal{P}_S = \{\mathsf{out}(c, [u]_i)^i.Q\} \uplus \mathcal{Q}_S$, $\mathcal{P}_S' = \{Q\} \uplus \mathcal{Q}_S$, and $\Phi_S' = \Phi_S \cup \{w_n \rhd [u]_i\sigma_S\}$.

Furthermore, we have that $\ell = \texttt{new}\ w_n.\texttt{out}(c, w_n)$, $c \notin \mathcal{E}$, and $n = |\Phi_S| + 1$. Lastly, since S is issued from $(\mathcal{E}; [\![P_0]\!]; \emptyset; \emptyset)$, we have that $\sigma_S \vDash \mathsf{test}_i([u]_i)$.

By hypothesis, we have that $D = \delta(S)$. Hence, we have that:

$$D = (\mathcal{E}; \{\texttt{out}(c, \delta_\gamma([u]_i)).\delta(Q)\} \uplus \delta(\mathcal{Q}_S); \Phi_D; \sigma_D)$$

with $\Phi_D{\downarrow} = \delta(\Phi_S{\downarrow})$, $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$, and $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$.

Hence, we have that $D \xrightarrow{\texttt{new}\ w_n.\texttt{out}(c,w_n)} D'$ where

$$D' = (\mathcal{E}; \delta(Q) \uplus \delta(\mathcal{Q}_S); \Phi_D \cup \{w_n \rhd \delta_\gamma([u]_i)\sigma_D\}; \sigma_D).$$

Hence, we have that $B \xrightarrow{\texttt{new}\ w_n.\texttt{out}(c,w_n)}_{\mathsf{bi}} B'$ with $\mathsf{fst}(B') = S'$ and $\mathsf{snd}(B') = D'$. It remains to show that $D' = \delta(S')$, i.e.

$$(\delta_\gamma([u]_i)\sigma_D){\downarrow} = \delta_\gamma([u]_i\sigma_S{\downarrow}).$$

Since $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$, we have that:

$$(\delta_\gamma([u]_i)\sigma_D){\downarrow} = (\delta_\gamma([u]_i)\delta(\sigma_S{\downarrow})){\downarrow}$$

Let γ' be equal to α if $\gamma = \beta$, and equal to β if $\gamma = \alpha$. Each variable that occurs in $[u]_i$ also occurs in $\mathrm{dom}(\sigma_S)$ and such a variable is either colored with a color in γ, or an assignation variable $z_j^{\gamma'}$. Thus, we have that $\delta_\gamma([u]_i)$ only contains variables that are colored with a color in γ. Hence, we have that

$$(\delta_\gamma([u]_i)\delta(\sigma_S{\downarrow})){\downarrow} = (\delta_\gamma([u]_i)\delta_\gamma(\sigma_S{\downarrow})){\downarrow}$$

Relying on Lemma 7 (note that $\sigma_S{\downarrow} \vDash \mathsf{test}_i([u]_i)$), we have that:

$$\begin{aligned}(\delta_\gamma([u]_i)\sigma_D){\downarrow} &= (\delta_\gamma([u]_i)\delta_\gamma(\sigma_S{\downarrow})){\downarrow}\\ &= \delta_\gamma([u]_i(\sigma_S{\downarrow})){\downarrow}\\ &= \delta_\gamma([u]_i(\sigma_S{\downarrow}){\downarrow})\\ &= \delta_\gamma([u]_i\sigma_S{\downarrow})\end{aligned}$$

Case of the rule IN. In such a case, we have that $\mathcal{E}'_S = \mathcal{E}_S$, $\Phi'_S = \Phi_S$, $\mathcal{P}_S = \{\texttt{in}(c, x)^i.Q\} \uplus \mathcal{Q}_S$, $\mathcal{P}'_S = \{Q\} \uplus \mathcal{Q}_S$, $\sigma'_S = \sigma_S \cup \{x \mapsto M\Phi_S\}$, and $\ell = \texttt{in}(c, M)$ with $c \notin \mathcal{E}_S$, $fv(M) \subseteq \mathrm{dom}(\Phi_S)$ and $fn(M) \cap \mathcal{E}_S = \emptyset$.

By hypothesis, we have that $D = \delta(S)$. Hence, we have that:

$$D = (\mathcal{E}; \{\texttt{in}(c, x).\delta(Q)\} \uplus \delta(\mathcal{Q}_S); \Phi_D; \sigma_D)$$

with $\Phi_D{\downarrow} = \delta(\Phi_S{\downarrow})$, $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$. Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$.

Hence, we have that $D \xrightarrow{\texttt{in}(c,M)} D'$ where

$$D' = (\mathcal{E}; \delta(Q) \uplus \delta(\mathcal{Q}_S); \Phi_D; \sigma_D \cup \{x \mapsto M\Phi_D\}).$$

Hence, we have that $B \xrightarrow{\texttt{in}(c,M)}_{\mathsf{bi}} B'$ with $\mathsf{fst}(B') = S'$ and $\mathsf{snd}(B') = D'$. It remains to show that $D' = \delta(S')$, i.e.

$$(M\Phi_D){\downarrow} = \delta_\gamma(M\Phi_S{\downarrow}).$$

By hypothesis, we know that D_0 does not reveal the values of its assignment variables w.r.t. $(\rho_\alpha, \rho_\beta)$. Hence, for all assignment variable x of color α (resp. β) in $\mathrm{dom}(\sigma_D)$, for all $k \in \{k, \mathsf{pk}(k), \mathsf{vk}(k) \mid k = x\sigma_D \vee k = x\rho_\alpha \text{ (resp. } x\rho_\beta)\}$, k is not deducible in $\mathtt{new}\ \mathcal{E}.\Phi_D$. We denote K this set.
Let $K_S = \{t, \mathsf{pk}(t), \mathsf{vk}(t) \mid t \in \mathrm{dom}(\rho_\alpha^+) \cup \mathrm{dom}(\rho_\beta^+),\ t \text{ ground}\}$ We know that $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$, and by definition of ρ_α^+ and ρ_β^+, we have that $K = \delta_\alpha(K_S) \cup \delta_\beta(K_S)$. We have also that $\Phi_D{\downarrow} = \delta(\Phi_S{\downarrow})$. Hence, we deduce that $\mathtt{new}\ \mathcal{E}.\delta(\Phi_S{\downarrow}) \not\vdash k$ for any $k \in \delta_\alpha(K_S) \cup \delta_\beta(K_S)$ This allow us to apply Lemma 16 and thus to obtain that:

$$(M\Phi_D){\downarrow} = (M(\Phi_D{\downarrow})){\downarrow} = (M\delta(\Phi_S{\downarrow})){\downarrow} = \delta_\gamma(M\Phi_S{\downarrow}).$$

Case of the rule THEN. In such a case, we have that $\mathcal{E}'_S = \mathcal{E}_S$, $\Phi'_S = \Phi_S$, $\sigma'_S = \sigma_S$, $\mathcal{P}_S = \{P_S\} \uplus \mathcal{Q}_S$, and $\mathcal{P}'_S = \{P'_S\} \uplus \mathcal{Q}_S$ where P_S and P'_S are as follows:

- *Case a: a test before an output.*

$$\begin{array}{l} P_S = \mathtt{if}\ \mathsf{test}_i([v]_i)\ \mathtt{then}\ \mathtt{out}(c, [v]_i)^i.Q_S \\ P'_S = \mathtt{out}(u, [v]_i)^i.Q_S \\ \sigma_S \vDash \mathsf{test}_i([v]_i) \end{array}$$

  for some $i \in \{1, \ldots, p\}$.
- *Case b: a test before an assignation.*

$$\begin{array}{l} P_S = \mathtt{if}\ \mathsf{test}_i([v]_i)\ \mathtt{then}\ [z := [v]_i]^i.Q_S \\ P'_S = \{[z := [v]_i]^i.Q_S \\ \sigma_S \vDash \mathsf{test}_i([v]_i) \end{array}$$

  for some $i \in \{1, \ldots, p\}$.
- *Case c: a test before a conditional.*

$$\begin{array}{l} P_S = \mathtt{if}\ \mathsf{test}_i([\varphi]_i)\ \mathtt{then}\ (\mathtt{if}\ [\varphi]_i\ \mathtt{then}\ Q_S^1\ \mathtt{else}\ Q_S^2) \\ P'_S = \mathtt{if}\ [\varphi]_i\ \mathtt{then}\ Q_S^1\ \mathtt{else}\ Q_S^2 \\ \sigma_S \vDash \mathsf{test}_i([\varphi]_i) \end{array}$$

  for some $i \in \{1, \ldots, p\}$.
- *Case d: a test of a conditional.*

$$\begin{array}{l} P_S = \mathtt{if}\ [\varphi]_i\ \mathtt{then}\ Q_S^1\ \mathtt{else}\ Q_S^2 \\ P'_S = Q_S^1 \\ \sigma_S \vDash [\varphi]_i \text{ and } \sigma_S \vDash \mathsf{test}_i([\varphi]_i) \end{array}$$

  for some $i \in \{1, \ldots, p\}$.

Each case can be handled in a similar way. Note that we rely on Corollary 1 instead of Lemma 8 to establish the result in *Case d.* We assume that we are in the first case. Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. By hypothesis, we have that $D = \delta(S)$. Hence, we have that D is equal to

$$\begin{array}{l}(\mathcal{E}; \{\mathtt{if}\ \mathsf{test}_i(\delta_\gamma([v]_i))\ \mathtt{then} \\ \qquad \mathtt{out}(c, \delta_\gamma([v]_i)).\delta(Q_S)\} \uplus \mathcal{Q}_S; \Phi_D; \sigma_D)\end{array}$$

with $\Phi_D{\downarrow} = \delta(\Phi_S{\downarrow})$, and $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$.

Since $\sigma_S \vDash \mathsf{test}_i([v]_i)$, we have also that $(\sigma_S{\downarrow}) \vDash \mathsf{test}_i([v]_i)$. Thanks to Lemma 8, we deduce that $\delta_\gamma(\sigma_S{\downarrow}) \vDash \mathsf{test}_i(\delta_\gamma([v]_i))$. Actually, each variable that occurs in $\mathsf{test}_i(\delta_\gamma([v]_i))$ is a variable that occurs in $\mathrm{dom}(\sigma_S)$ and such a variable is necessarily colored with a color in γ. Hence, we have also that:

$$\delta(\sigma_S{\downarrow}) \vDash \mathsf{test}_i(\delta_\gamma([v]_i)).$$

Hence, we have that $D \xrightarrow{\tau} D'$ where

$$D' = (\mathcal{E}; \{\mathtt{out}(u, \delta_\gamma([v]_i)).\delta(Q_S)\} \uplus \delta(\mathcal{Q}_S); \Phi_D; \sigma_D).$$

Hence, we have that $B \xrightarrow{\tau}_{\mathsf{bi}} B'$ with $\mathsf{fst}(B') = S'$ and $\mathsf{snd}(B') = D'$. We also have that $D' = \delta(S')$.

Case of the rule ELSE. This case is similar to the previous one.

Case of the rule ASSGN. In such a case, we have that $\mathcal{E}'_S = \mathcal{E}_S$, $\Phi'_S = \Phi_S$, $\mathcal{P}_S = \{[x := [v]_i].Q\} \uplus \mathcal{Q}_S$, $\mathcal{P}'_S = \{Q\} \uplus \mathcal{Q}_S$, $\sigma'_S = \sigma_S \cup \{x \mapsto [v]_i\sigma_S\}$, and $\ell = \tau$. Lastly, since S is issued from $(\mathcal{E}; [\![P_0]\!]; \emptyset; \emptyset)$, we have that $\sigma_S \vDash \mathsf{test}_i([v]_i)$.

By hypothesis, we have that $D = \delta(S)$. Hence, we have that:

$$D = (\mathcal{E}; [x := \delta([v]_i)].\delta(Q) \uplus \delta(\mathcal{Q}_S); \Phi_D; \sigma_D)$$

with $\Phi_D{\downarrow} = \delta(\Phi_S{\downarrow})$, $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$, and $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$.

Hence, we have that $D \xrightarrow{\tau} D'$ where

$$D' = (\mathcal{E}; \delta(Q) \uplus \delta(\mathcal{Q}_S); \Phi_D; \sigma_D \cup \{x \mapsto \delta_\gamma([v]_i)\sigma_D\}).$$

Hence, we have that $B \xrightarrow{\tau} B'$ with $\mathsf{fst}(B') = S'$ and $\mathsf{snd}(B') = D'$. It remains to show that $D' = \delta(S')$, i.e.

$$(\delta_\gamma([v]_i)\sigma_D){\downarrow} = \delta_\gamma([v]_i\sigma_S{\downarrow}).$$

This can be done as in the case of the rule OUT-T.

Case of the rule COMM. In such a case, we have that $\mathcal{E}'_S = \mathcal{E}_S$, $\Phi'_S = \Phi_S$, $\mathcal{P}_S = \{\mathtt{out}(c, [u]_i)^i.Q_1; \mathtt{in}(c, x)^{i'}.Q_2\} \uplus \mathcal{Q}_S$, $\sigma'_S = \sigma_S \cup \{x \mapsto [u]_i\sigma_S\}$, and $\ell = \tau$. Lastly, since S is issued from $(\mathcal{E}; [\![P_0]\!]; \emptyset; \emptyset)$, we have that $\sigma_S \vDash \mathsf{test}_i([u]_i)$.

By hypothesis, we have that $D = \delta(S)$. Hence, we have that D is equal to

$$(\mathcal{E}; \{\mathtt{out}(c, \delta_\gamma([u]_i)).\delta(Q_1); \mathtt{in}(c, x).\delta(Q_2)\} \uplus \delta(\mathcal{Q}_S); \Phi_D; \sigma_D)$$

with $\Phi_D{\downarrow} = \delta(\Phi_S{\downarrow})$, $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$.

Let $\gamma, \gamma' \in \{\alpha, \beta\}$ such that $i \in \gamma$, and $i' \in \gamma'$. Hence, we have that $D \xrightarrow{\tau} D'$ where D' is equal to:

$$(\mathcal{E}; \{\delta(Q_1); \delta(Q_2)\} \uplus \delta(\mathcal{Q}_S); \Phi_D; \sigma_D \cup \{(x \mapsto \delta_\gamma([u]_i)\sigma_D)^{i'}\}).$$

Hence, we have that $B \xrightarrow{\tau} B'$ for some biprocess B' such that $\mathsf{fst}(B') = S'$ and $\mathsf{snd}(B') = D'$. It remains to show that $D' = \delta(S')$, i.e.

$$(\delta_\gamma([u]_i)\sigma_D){\downarrow} = \delta_{\gamma'}(([u]_i\sigma_S){\downarrow})$$

If $\gamma = \gamma'$, then this can be done as in the previous cases.

Otherwise, since names can only be shared through assignments, and assignments only concern variables/terms of base type, we necessarily have that $c \notin \mathcal{E}$. Hence, we have that $S \xrightarrow{\nu w_n.\mathtt{out}(c,w_n)} S_{\mathtt{out}}$ where:

$$S_{\mathtt{out}} = (\mathcal{E}; \{Q_1; \mathtt{in}(c,x).Q_2\} \uplus \mathcal{Q}_S; \Phi_S \cup \{w_n \rhd [u]_i\sigma_S\}; \sigma_S)$$

Note that $(\rho_\alpha, \rho_\beta)$ is still compatible with $S_{\mathtt{out}}$. We would like to apply Lemma 16 with $M = w_n$ on the frame of $S_{\mathtt{out}}$ which requires an hypothesis of non deductibility of the shared key. For these, we will rely on our hypothesis that D_0 does not reveal the values of its assignments w.r.t. $(\rho_\alpha, \rho_\beta)$:

Let $\Phi'_S = \Phi_S \cup \{w_n \rhd [u]_i\sigma_S\}$. We already proved our induction result for the rule OUT-T. Hence, we deduce that there exists $D_{\mathtt{out}}$ such that $D \xrightarrow{\nu w_n.\mathtt{out}(c,w_n)} D_{\mathtt{out}}$ where $D_{\mathtt{out}} = (\mathcal{E}; \mathcal{P}_{\mathtt{out}}; \Phi'_D; \sigma_D)$, $\Phi'_D = \Phi_D \cup \{w_n \rhd \delta_\gamma([u]_i)\sigma_D\}$. Moreover, it implies that $\Phi'_D{\downarrow} = \delta(\Phi'_S{\downarrow}\})$ and $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$. As mentioned, by hypothesis, we know that D_0 does not reveal the values of its assignments w.r.t. $(\rho_\alpha, \rho_\beta)$. Hence, for all assignment variable x of color α (resp. β) in $\mathrm{dom}(\sigma_D)$, for all $k \in \{k, \mathsf{pk}(k), \mathsf{vk}(k) \mid k = x\sigma_D \vee k = x\rho_\alpha \text{ (resp. } x\rho_\beta)\}$, k is not deducible in $\mathtt{new}\ \mathcal{E}.\Phi'_D$. We denote by K such a set.
Let $K_S = \{t, \mathsf{pk}(t), \mathsf{vk}(t) \mid t \in \mathrm{dom}(\rho_\alpha^+) \cup \mathrm{dom}(\rho_\beta^+),\ t \text{ ground}\}$. Since $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$, and by definition of ρ_α^+ and ρ_β^+, we deduce that $K = \delta_\alpha(K_S) \cup \delta_\beta(K_S)$. Moreover, we have that $\Phi'_D{\downarrow} = \delta(\Phi'_S{\downarrow})$. Hence, we deduce that $\mathtt{new}\ \mathcal{E}.\delta(\Phi'_S{\downarrow}) \not\vdash k$ for any $k \in \delta_\alpha(K_S) \cup \delta_\beta(K_S)$. This allow us to apply Lemma 16 with $M = w_n$ and so we deduce that $\delta_\gamma([u]_i\sigma_S{\downarrow}) = \delta_{\gamma'}([u]_i\sigma_S{\downarrow})$. Hence, we can conclude as in the previsous case.

Case of the rule PAR. It is easy to see that the result holds for this case.

Note that the rules NEW and REPL can not be triggered since the processes under study do not contain bounded names and replication.

Proposition 2. *Let P_0 be a plain colored process without replication and such that $bn(P_0) = fv(P_0) = \emptyset$. Let B_0 be an extended colored biprocess such that:*

- $S_0 = (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta \uplus \mathcal{E}_0; [\![P_0]\!]; \emptyset; \emptyset) \stackrel{\mathsf{def}}{=} \mathsf{fst}(B_0)$,
- $D_0 = (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta \uplus \mathcal{E}_0; P'_0; \emptyset; \emptyset) \stackrel{\mathsf{def}}{=} \mathsf{snd}(B_0)$, *and*
- $D_0 = \delta^{\rho_\alpha^+, \rho_\beta^+}(S_0)$ *for some* $(\rho_\alpha, \rho_\beta)$.

– D_0 *does not reveal the value of its assignments w.r.t.* $(\rho_\alpha, \rho_\beta)$.

For any extended process $D = (\mathcal{E}_D; \mathcal{P}_D; \Phi_D; \sigma_D)$ *such that* $D_0 \stackrel{\mathsf{tr}}{\Rightarrow} D$ *with* $(\rho_\alpha, \rho_\beta)$ *compatible with* D*, there exists a biprocess* B *and an extended process* $S = (\mathcal{E}_S; \mathcal{P}_S; \Phi_S; \sigma_S)$ *such that* $B_0 \stackrel{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} B$*, ,*$\mathsf{fst}(B) = S$*,* $\mathsf{snd}(B) = D$*, and* $D = \delta(S)$*.*

Proof. We show the result by induction on the length of the derivation. The base case when $D_0 = D$ is trivial. We simply conclude by considering $B = B_0$, and $S = S_0$. Now, we assume that $D_0 \stackrel{\mathsf{tr}'}{\Rightarrow} D'$ such that $(\rho_\alpha, \rho_\beta)$ is compatible with D'. This means that there exist D, tr, and ℓ such that:

$$D_0 \stackrel{\mathsf{tr}}{\Rightarrow} D \xrightarrow{\ell} D' \text{ with } \mathsf{tr}' = \mathsf{tr} \cdot \ell$$

Note that we necessarily have that $(\rho_\alpha, \rho_\beta)$ is compatible with D.

By induction hypothesis, we have that there exists an extended biprocess B and an extended process S such that $\mathsf{fst}(B) = S$, $\mathsf{snd}(B) = D$, $B_0 \stackrel{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} B$, and $D = \delta(S)$. We show the result by case analysis on the rule involved in $D \xrightarrow{\ell} D'$. Let $D = (\mathcal{E}_D; \mathcal{P}_D; \Phi_D; \sigma_D)$ and $D' = (\mathcal{E}'_D; \mathcal{P}'_D; \Phi'_D; \sigma'_D)$. First, note that since D is issued from $D_0 = \delta(S_0)$ and $S_0 = (\mathcal{E}; [\![P_0]\!]; \emptyset; \emptyset)$, we know that terms invovled in D are tagged and obtained through the δ transformation.

Case of the rule Out-T. In such a case, we have that $\mathcal{E}'_D = \mathcal{E}_D$, $\sigma'_D = \sigma_D$, $\mathcal{P}_D = \{\mathtt{out}(c, \delta([v]_i)).\delta(Q_S)\} \uplus \delta(\mathcal{Q}_S)$, $\mathcal{P}'_D = \{\delta(Q_D)\} \uplus \delta(\mathcal{Q}_D)$, and $\Phi'_D = \Phi_D \cup \{w_n \rhd \delta_\gamma([v]_i)\sigma_D\}$ with $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. Furthermore, we have that $\ell = \mathtt{new}\, w_n.\mathtt{out}(c, w_n)$, $c \notin \mathcal{E}_D$, and $n = |\Phi_D| + 1$. We have also

$$S = (\mathcal{E}; \{\mathtt{out}(c, [v]_i).Q_S\} \uplus \mathcal{Q}_S); \Phi_S; \sigma_S)$$

with $\Phi_D{\downarrow} = \delta(\Phi_S{\downarrow})$, and $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$. Hence, we have that $S \xrightarrow{\mathtt{new}\, w_n.\mathtt{out}(c, w_n)} S'$ where

$$S' = (\mathcal{E}; Q_S \uplus \mathcal{Q}_S; \Phi_S \cup \{w_n \rhd [v]_i\sigma_S\}; \sigma_S).$$

Hence, we have that $B \xrightarrow{\mathtt{new}\, w_n.\mathtt{out}(c, w_n)}_{\mathsf{bi}} B'$ with $\mathsf{fst}(B') = S'$ and $\mathsf{snd}(B') = D'$. It remains to show that $D' = \delta(S')$, i.e.

$$(\delta_\gamma([v]_i)\sigma_D){\downarrow} = \delta_\gamma([v]_i\sigma_S{\downarrow})$$

Since D is issued from $(\mathcal{E}; \delta([\![P_0]\!]); \emptyset; \emptyset)$ and $B_0 \stackrel{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} B$, we have that $\sigma_D \vDash \mathsf{test}_i(\delta_\gamma([v]_i))$ and $\sigma_S \vDash \mathsf{test}_i([v]_i)$.

Since $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$, we have that:

$$(\delta_\gamma([v]_i)\sigma_D){\downarrow} = (\delta_\gamma([v]_i)\delta(\sigma_S{\downarrow})){\downarrow}$$

Let γ' be equal to α if $\gamma = \beta$, and equal to β if $\gamma = \alpha$. Each variable that occurs in $[v]_i$ also occurs in $\mathrm{dom}(\sigma_S)$ and such a variable is either colored with a color in γ, or an assignation variable $z_j^{\gamma'}$. Thus, we have that $\delta_\gamma([v]_i)$ only contains variables that are colored with a color in γ. Hence, we have that

$$(\delta_\gamma([v]_i)\delta(\sigma_S{\downarrow})){\downarrow} = (\delta_\gamma([v]_i)\delta_\gamma(\sigma_S{\downarrow})){\downarrow}$$

Relying on Lemma 7 (note that $\sigma_S\!\downarrow \vDash \mathsf{test}_i([v]_i)$), we have that:

$$\begin{aligned}(\delta_\gamma([v]_i)\sigma_D)\!\downarrow &= (\delta_\gamma([v]_i)\delta_\gamma(\sigma_S\!\downarrow))\!\downarrow\\ &= \delta_\gamma([v]_i(\sigma_S\!\downarrow))\!\downarrow\\ &= \delta_\gamma([v]_i(\sigma_S\!\downarrow)\!\downarrow)\\ &= \delta_\gamma([v]_i\sigma_S\!\downarrow)\end{aligned}$$

Case of the rule IN. In such a case, we have that $\mathcal{E}'_D = \mathcal{E}_D$, $\Phi'_D = \Phi_D$, $\mathcal{P}_D = \{\in (c,x)^i.\delta(Q_S\} \uplus \delta(\mathcal{Q}_S)$, $\mathcal{P}'_D = \{\delta(Q_S)\} \uplus \delta(\mathcal{Q}_S)$, $\sigma'_D = \sigma_D \cup \{x \mapsto M\Phi_D\}$, and $\ell = \mathtt{in}(c,M)$ with $c \notin \mathcal{E}_D$, $fv(M) \subseteq \mathrm{dom}(\Phi_D)$, and $fn(M) \cap \mathcal{E}_D = \emptyset$. Moreover, we have that:

$$S = (\mathcal{E}; \{\mathtt{in}(c,x).Q_S\} \uplus \mathcal{Q}_S; \Phi_S; \sigma_S)$$

with $\Phi_D\!\downarrow = \delta(\Phi_S\!\downarrow)$, and $\sigma_D\!\downarrow = \delta(\sigma_S\!\downarrow)$. Let $\gamma \in \{\alpha,\beta\}$ such that $i \in \gamma$.

Hence, we have that $S \xrightarrow{\mathtt{in}(c,M)} S'$ where

$$S' = (\mathcal{E}; \{Q_S\} \uplus \mathcal{Q}_S; \Phi_S; \sigma_S \cup \{x \mapsto M\Phi_S).$$

Hence, we have that $B \xrightarrow{\mathtt{in}(c,M)}_{\mathsf{bi}} B'$ with $\mathsf{fst}(B') = S'$, and $\mathsf{snd}(B') = D'$. It remains to show that $D' = \delta(S')$, i.e.

$$(M\Phi_D)\!\downarrow = \delta_\gamma(M\Phi_S\!\downarrow).$$

By hypothesis, we know that D_0 does not reveal the value of its assignments w.r.t. $(\rho_\alpha, \rho_\beta)$. Since $\Phi_D\!\downarrow = \delta(\Phi_S\!\downarrow)$ and $\sigma_D\!\downarrow = \delta(\sigma_S\!\downarrow)$. Hence, by following the definition of ρ^+_α and ρ^+_β, we deduce that the hypothesis of Lemma 16 are satisfied. Hence, by relying on it, we have that:

$$(M\Phi_D)\!\downarrow = (M(\Phi_D\!\downarrow))\!\downarrow = (M\delta(\Phi_S\!\downarrow))\!\downarrow = \delta_\gamma(M\Phi_S\!\downarrow).$$

Case of the rule THEN. In such a case, we have that $\mathcal{E}'_D = \mathcal{E}_D$, $\Phi'_D = \Phi_D$, $\sigma'_D = \sigma_D$, $\mathcal{P}_D = \{P_D\} \uplus \mathcal{Q}_D$, and $\mathcal{P}'_D = \{P'_D\} \uplus \mathcal{Q}_D$ where P_D and P'_D are as follows:

– *Case a: a test before an output.*

$$\begin{aligned}&P_D = \mathtt{if}\,\mathsf{test}_i(\delta_\gamma([v]_i))\,\mathtt{then}\,\mathtt{out}(c,\delta_\gamma([v]_i))^i.\delta(Q_S)\\ &P'_D = \mathtt{out}(u,\delta_\gamma([v]_i))^i.\delta(Q_S)\\ &\sigma_D \vDash \mathsf{test}_i(\delta_\gamma([v]_i))\end{aligned}$$

for some $i \in \{1,\dots,p\}$ and $\gamma \in \{\alpha,\beta\}$ such that $i \in \gamma$.

– *Case b: a test before an assignation.*

$$\begin{aligned}&P_D = \mathtt{if}\,\mathsf{test}_i(\delta_\gamma([v]_i))\,\mathtt{then}\,[z := \delta_\gamma([v]_i)]^i.\delta(Q_S)\\ &P'_D = \{[z := \delta_\gamma([v]_i)]^i.\delta(Q_S)\\ &\sigma_D \vDash \mathsf{test}_i(\delta_\gamma([v]_i))\end{aligned}$$

for some $i \in \{1,\dots,p\}$ and $\gamma \in \{\alpha,\beta\}$ such that $i \in \gamma$.

– *Case c: a test before a conditional.*

$$\begin{array}{l} P_D = \mathtt{if}\,\mathsf{test}_i(\delta_\gamma([\varphi]_i))\,\mathtt{then} \\ \qquad\quad (\mathtt{if}\,[\varphi]_i\,\mathtt{then}\,\delta(Q_S^1)\,\mathtt{else}\,\delta(Q_S^2)) \\ P'_D = \mathtt{if}\,[\varphi]_i\,\mathtt{then}\,\delta(Q_S^1)\,\mathtt{else}\,\delta(Q_S^2) \\ \sigma_D \vDash \mathsf{test}_i(\delta_\gamma([\varphi]_)i) \end{array}$$

for some $i \in \{1, \dots, p\}$ and $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$.

– *Case d: a test of a conditional.*

$$\begin{array}{l} P_D = \mathtt{if}\,\delta_\gamma([\varphi]_i)\,\mathtt{then}\,\delta(Q_S^1)\,\mathtt{else}\,\delta(Q_S^2) \\ P'_D = \delta(Q_S^1) \\ \sigma_D \vDash \delta_\gamma([\varphi]_i) \text{ and } \sigma_D \vDash \mathsf{test}_i(\delta_\gamma([\varphi]_i)) \end{array}$$

for some $i \in \{1, \dots, p\}$ and $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$.

Each case can be handled in a similar way. Note that we rely in addition on Corollary 1 instead of Lemma 8 to establish the result in *case d*. We assume that we are in the first case. Let $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. We have that S is equal to

$$(\mathcal{E}; \{\mathtt{if}\,\mathsf{test}_i([v]_i)\,\mathtt{then}\,\mathtt{out}(c, [v]_i)^i.Q_S\} \uplus \mathcal{Q}_S; \Phi_S; \sigma_S)$$

with $\Phi_D{\downarrow} = \delta(\Phi_S{\downarrow})$, and $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$.

Since $\sigma_D \vDash \mathsf{test}_i(\delta_\gamma([v]_i))$, we have $(\sigma_D{\downarrow}) \vDash \mathsf{test}_i(\delta_\gamma([v]_i))$, and thus $\delta(\sigma_S{\downarrow}) \vDash \mathsf{test}_i(\delta_\gamma([v]_i))$. As in the previous cases, we deduce that $\delta_\gamma(\sigma_S{\downarrow}) \vDash \mathsf{test}_i(\delta_\gamma([v]_i))$. Thanks to Lemma 8, we deduce that $\sigma_S{\downarrow} \vDash \mathsf{test}_i([v]_i)$. Hence, we have that $S \xrightarrow{\tau} S'$ where

$$S' = (\mathcal{E}; \{\mathtt{out}(u, [v]_i).Q_S\} \uplus \mathcal{Q}_S; \Phi_S; \sigma_S).$$

Hence, we have that $B \xrightarrow{\tau}_{\mathsf{bi}} B'$ with $\mathsf{fst}(B') = S'$, and $\mathsf{snd}(B') = D'$. We also have that $D' = \delta(S')$.

Case of the rule Else. This case is similar to the previous one.

Case of the rule Assgn. In such a case, we have that $\mathcal{E}'_D = \mathcal{E}_D$, $\Phi'_D = \Phi_D$, $\mathcal{P}_D = \{[x := \delta_\gamma([v]_i)].\delta(Q)\} \uplus \delta(\mathcal{Q}_S)$, $\mathcal{P}'_D = \{\delta(Q)\} \uplus \delta(\mathcal{Q}_S)$, $\sigma'_D = \sigma_D \cup \{x \mapsto \delta_\gamma([v]_i)\sigma_D\}$, and $\ell = \tau$ where $\gamma \in \{\alpha, \beta\}$ with $i \in \gamma$. We have also that $\sigma_D \vDash \mathsf{test}_i(\delta([v]_i))$ and $\sigma_S \vDash \mathsf{test}_i([v]_i)$. Hence, we have that:

$$S = (\mathcal{E}; \{[x := [v]_i].Q\} \uplus \mathcal{Q}_S; \Phi_S; \sigma_S)$$

with $\Phi_D{\downarrow} = \delta(\Phi_S{\downarrow})$, and $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$.

Hence, we have that $S \xrightarrow{\tau} S'$ where:

$$S' = (\mathcal{E}; \{Q\} \uplus \mathcal{Q}_S; \Phi_S; \sigma_S \cup \{x \mapsto [v]_i\sigma_S\}).$$

Hence, we have that $B \xrightarrow{\tau}_{\mathsf{bi}} B'$ with $\mathsf{fst}(B') = S'$ and $\mathsf{snd}(B') = D'$. It remains to show that $D' = \delta(S')$, i.e.

$$(\delta_\gamma([v]_i)\sigma_D){\downarrow} = \delta_\gamma([v]_i\sigma_S{\downarrow}).$$

This can be done as in the case of the rule Out-T.

Case of the rule Comm. In such a case, $\mathcal{P}_D = \{\mathtt{out}(c, \delta_\gamma([u]_i))^i.\delta(Q_1); \mathtt{in}(c,x)^{i'}.\delta(Q_2)\} \uplus \delta(\mathcal{Q}_S)$, $\mathcal{E}'_D = \mathcal{E}_D$, $\Phi'_D = \Phi_D$, $\sigma'_D = \sigma_D \cup \{x \mapsto \delta_\gamma([u]_i)\sigma_D\}$, and $\ell = \tau$. Moreover, we have that $\sigma_D \vDash \delta_\gamma(\mathsf{test}_i([u]_i))$ and $\sigma_S \vDash \mathsf{test}_i([v]_i)$ where $\gamma \in \{\alpha, \beta\}$ such that $i \in \gamma$. Hence, we have that S is equal to

$$(\mathcal{E}; \{\mathtt{out}(c, [u]_i).Q_1; \mathtt{in}(c,x).Q_2\} \uplus \mathcal{Q}_S; \Phi_S; \sigma_S)$$

with $\Phi_D{\downarrow} = \delta(\Phi_S{\downarrow})$, and $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$.

Let $\gamma' \in \{\alpha, \beta\}$ such that $i' \in \gamma'$. Hence, we have that $S \xrightarrow{\tau} S'$ where S' is equal to:

$$(\mathcal{E}; \{Q_1; Q_2\} \uplus \mathcal{Q}_S; \Phi_S; \sigma_S \cup \{(x \mapsto [u]_i\sigma_S)^{i'}\}).$$

Hence, we have that $B \xrightarrow{\tau} B'$ for some biprocess B' such that $\mathsf{fst}(B') = S'$ and $\mathsf{snd}(B') = D'$. It remains to show that $D' = \delta(S')$, i.e.

$$(\delta_\gamma([u]_i)\sigma_D){\downarrow} = \delta_{\gamma'}([u]_i\sigma_S{\downarrow})$$

If $\gamma = \gamma'$, then this can be done as in the previous cases. Otherwise, since names can only be shared through assignations, and assignations only concern variables/terms of base type, we necessarily have that $c \notin \mathcal{E}$. Hence, we have that $D \xrightarrow{\nu w_n.\mathtt{out}(c,w_n)} D_{\mathtt{out}}$ where $D_{\mathtt{out}}$ is equal to:

$$(\mathcal{E}_D; \{\delta(Q_1); \mathtt{in}(c,x).\delta(Q_2)\} \uplus \mathcal{Q}_S; \Phi_D \cup \{w_n \rhd \delta_\gamma([u]_i)\sigma_D\}; \sigma_D)$$

Note that $(\rho_\alpha, \rho_\beta)$ is still compatible with $D_{\mathtt{out}}$. We would like to apply Lemma 16 with $M = w_n$ on the frame of $D_{\mathtt{out}}$ which requires an hypothesis of non deductibility of the shared key. For these, we will rely on our hypothesis that D_0 does not reveal the values of his assignment variables w.r.t. $(\rho_\alpha, \rho_\beta)$:

Let $\Phi'_D = \Phi_D \cup \{w_n \rhd \delta_\gamma([u]_i)\sigma_D\}$. We already proved our induction result for the rule Out-T. Hence, we deduce that there exists $S_{\mathtt{out}}$ such that $S \xrightarrow{\nu w_n.\mathtt{out}(c,w_n)} S_{\mathtt{out}}$ where $S_{\mathtt{out}} = (\mathcal{E}; \mathcal{P}'_S; \Phi'_S; \sigma_S)$, $\Phi'_S = \Phi_S \cup \{w_n \rhd [u]_i)\sigma_S\}$. Moreover, it implies that $\Phi'_D{\downarrow} = \delta(\Phi'_S{\downarrow}\})$ and $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$. As mentioned, by hypothesis, we know that D_0 does not reveal the values of its assignments w.r.t. $(\rho_\alpha, \rho_\beta)$. Hence, for all assignment variable x of color α (resp. β) in $\mathrm{dom}(\sigma_D)$, for all *key* $\in \{k, \mathsf{pk}(k), \mathsf{vk}(k) \mid k = x\sigma_D \vee k = x\rho_\alpha$ (resp. $x\rho_\beta$)$\}$, *key* is not deducible from $\mathtt{new}\ \mathcal{E}.\Phi'_D$. We denote K this set.
Let $K_S = \{t, \mathsf{pk}(t), \mathsf{vk}(t) \mid t \in \mathrm{dom}(\rho^+_\alpha) \cup \mathrm{dom}(\rho^+_\beta),\ t \text{ ground}\}$. Since $\sigma_D{\downarrow} = \delta(\sigma_S{\downarrow})$, and by definition of ρ^+_α and ρ^+_β, we deduce that $K = \delta_\alpha(K_S) \cup \delta_\beta(K_S)$. We have also that $\Phi'_D{\downarrow} = \delta(\Phi'_S{\downarrow})$. Hence, we can now apply Lemma 16 with $M = w_n$ and so we deduce that $\delta_\gamma([u]_i\sigma_S{\downarrow}) = \delta_{\gamma'}([u]_i\sigma_S{\downarrow})$. Hence, we can conclude as in the previous case.

Case of the rule Par. It is easy to see that the result holds for this case.

Note that the rules New and Repl can not be triggered since the processes under study do not contain bounded names and replication.

Theorem 5. *Let P be a plain colored process as described above, and B_0 be an extended colored biprocess such that:*

- $S_0 = (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta \uplus \mathcal{E}_0; [\![P]\!]; \emptyset; \emptyset) \overset{\mathsf{def}}{=} \mathsf{fst}(B_0)$,
- $D_0 = (\mathcal{E}_\alpha \uplus \mathcal{E}_\beta \uplus \mathcal{E}_0; P_D; \emptyset; \emptyset) \overset{\mathsf{def}}{=} \mathsf{snd}(B_0)$, *and*
- $P_D = \delta_{\rho_\alpha,\rho_\beta}([\![P]\!])$ *for some* $(\rho_\alpha, \rho_\beta)$ *compatible with* D_0, *and*
- D_0 *does not reveal its assignments w.r.t.* $(\rho_\alpha, \rho_\beta)$.

We have that:

1. *For any extended process* $S = (\mathcal{E}_S; \mathcal{P}_S; \Phi_S; \sigma_S)$ *such that* $S_0 \overset{\mathsf{tr}}{\Longrightarrow} S$ *with* $(\rho_\alpha, \rho_\beta)$ *compatible with* S, *there exists a biprocess* B *and an extended process* $D = (\mathcal{E}_D; \mathcal{P}_D; \Phi_D; \sigma_D)$ *such that* $B_0 \overset{\mathsf{tr}}{\Longrightarrow}_{\mathsf{bi}} B$, $\mathsf{fst}(B) = S$, $\mathsf{snd}(B) = D$, *and* $\mathtt{new}\,\mathcal{E}_S.\Phi_S \sim \mathtt{new}\,\mathcal{E}_D.\Phi_D$.
2. *For any extended process* $D = (\mathcal{E}_D; \mathcal{P}_D; \Phi_D; \sigma_D)$ *such that* $D_0 \overset{\mathsf{tr}}{\Longrightarrow} D$ *with* $(\rho_\alpha, \rho_\beta)$ *compatible with* D, *there exists a biprocess* B *and an extended process* $S = (\mathcal{E}_S; \mathcal{P}_S; \Phi_S; \sigma_S)$ *such that* $B_0 \overset{\mathsf{tr}}{\Longrightarrow}_{\mathsf{bi}} B$, $\mathsf{fst}(B) = S$, $\mathsf{snd}(B) = D$, *and* $\mathtt{new}\,\mathcal{E}_S.\Phi_S \sim \mathtt{new}\,\mathcal{E}_D.\Phi_D$.

Proof. We prove the two items separately.

1. The first item is actually a direct consequence of Proposition 1. We rely on Corollary 5 and the fact that $D = \delta(S)$ to establish that:

$$\mathtt{new}\,\mathcal{E}_S.\Phi_S \sim \mathtt{new}\,\mathcal{E}_D.\Phi_D.$$

2. The second item is actually a direct consequence of Proposition 2. We rely on Corollary 5, Corollary 4 and the fact that $D = \delta(S)$ to establish that:

$$\mathtt{new}\,\mathcal{E}_S.\Phi_S \sim \mathtt{new}\,\mathcal{E}_D.\Phi_D.$$

This concludes the proof of the theorem.

F Parallel composition

The goal of this section is to prove the results that relate to the parallel composition, that are Theorem 1 and 3. We prove a slightly improved version of Theorem 3 assuming that composition contexts may contain several holes. To prove these composition results, we will rely on Theorem 5, and for this we have to explain how to get rid of the replications, and the `new` instructions (see Section F.1). We have also to rewrite the process to ensure that names are shared via assignment variables only (see Section F.2).

F.1 Unfolding a biprocess

Given an extended process $A = (\mathcal{E}; \mathcal{P}; \Phi)$ where $\mathcal{P}$ may contain name restrictions and replications, the idea is to unfold the replications and to gather together all the restricted names in the set $\mathcal{E}$. Of course, it is not possible to apply such a transformation and to preserve the set of possible traces. However, given a specific trace issued from A, it is possible to compute an unfolding of A that will exhibit this specific trace. The converse is also true, any trace issued from an unfolding of A will correspond to a trace of A. Thus, the process A and all its possible unfoldings will exhibit exactly the same set of traces. We define this notion directly on biprocesses.

Definition 20. *Let $A = (\mathcal{E}; \mathcal{P}; \Phi)$ be an extended biprocess. We define the n^{th} unfolding of A, denote by $\mathsf{Unf}_n(A)$, the biprocess $(\mathcal{E} \uplus \mathcal{E}_n; \mathcal{P}_n; \Phi)$ obtained from A by replacing in $\mathcal{P}$ each instance of $!Q$ with n instances of Q (applying α-renaming to ensure name and variable distinctness), and then removing the* `new` *operations from the resulting process. These names are then put in the set $\mathcal{E}_n$ and added in the first component of the extended process.*

The link between an extended biprocess and its unfoldings is stated in Lemma 17.

Lemma 17. *Let $A = (\mathcal{E}; \mathcal{P}; \Phi)$ be an extended biprocess. The biprocess A is in diff-equivalence if, and only if, $\mathsf{Unf}_n(A)$ is in diff-equivalence for any $n \in \mathbb{N}$.*

F.2 Sharing names via assignments

In Theorem 5, one can note that processes may only share data through assignment variables. This is not a real limitation since a name that is shared via the composition context can be assigned to an assignment variable by one process and used by the other through the assignment variables. Below, we describe this transformation that actually preserves diff-equivalence of a biprocess.

Let $A = (\mathcal{E}; \mathcal{P}; \Phi)$ be an extended colored (with colors in $\{1, \ldots, p\} = \alpha \uplus \beta$) biprocess that does not contain any name restriction nor replication in $\mathcal{P}$. Let $K = k_1, \ldots, k_\ell$ be a sequence of names (of base type) in $\mathcal{E}$ that contains at least all the names occurring in both type of actions – in actions colored α as well as in actions colored β (intuitively $k_1, \ldots, k_\ell$ are the names shared by the two processes we want to compose). Since we work with a biprocess, we do this transformation simulatenously on both sides. We do this each time the transformation is required by one side of the biprocess. Actually, when we will apply this transformation, the right-hand side will correspond to the disjoint case, whereas the left-hand side will correspond to the shared case, and all the transformations will arise because of the left-hand side.

Let $Z = z_1^\alpha, \ldots, z_\ell^\alpha$ be a sequence of fresh variables, and $i \in \alpha$. We denote by $Ass^i_{Z:=\mathcal{K}}(A)$ the extended biprocess $(\mathcal{E}; P_{\mathsf{ass}}; \Phi)$ where P_{ass} is defined as follows:

$$P_{\mathsf{ass}} = [z_1^\alpha := k_1]^i. \ldots .[z_\ell^\alpha := k_\ell]^i.(|_{P \in \mathcal{P}} \; P\rho_\beta)$$

where ρ_β replaces each occurrence of the name k_j ($1 \leq j \leq \ell$) that occurs in an action β-colored by its associated assignment variable z_j^α ($1 \leq j \leq \ell$). Note that the replacement ρ_β will not affect the process corresponding to the disjoint case.

Note that in the definition above, the α-colored process will assign the shared names into assignment variables whereas the β-colored process will simply use those variables instead of the corresponding names. This choice is arbitrary and the roles played by α and β can be swapped. Again, this transformation preserves equivalence. This result is stated below in Lemma 18.

Lemma 18. *Let $A = (\mathcal{E}; \mathcal{P}; \Phi)$ and $Ass^i_{Z:=\mathcal{K}}(A)$ be two extended biprocesses as described above. We have that A is in diff-equivalence if, and only if, $Ass^i_{Z:=\mathcal{K}}(A)$ is in diff-equivalence.*

F.3 Composing trace equivalence

The theorem we want to prove is stated below. Note that, this theorem differs from the one stated in the main body of the paper since we work in a slightly more general setting.

We denote by $\Sigma_0^c = \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{pk}, \mathsf{vk}, \langle\ \rangle\}$, *i.e.* the constructors of the common signature Σ_0. We consider composition contexts that may contain several holes. They are formally defined as follows:

Definition 21. *A* composition context *C is defined by the following grammar where n is a name of base type.*

$$C, C_1, C_2 := \_ \ \mid\ \mathit{new}\ n.\ C\ \mid !C\ \mid\ C_1 | C_2$$

We only allow names of base type (typically keys) to be shared between processes through the composition context. In particular, they are not allowed to share a private channel even if each process can used its own private channels to communicate internally. We also suppose w.l.o.g. that names occurring in C are distinct. A composition context may contain several holes. We can index them to avoid confusion. We write $C[P_1, \ldots, P_\ell]$ (or shortly $C[\overline{P}]$) the process obtained by filling the i^{th} hole with the process P_i (or the i^{th} process of the sequence $\overline{P}$).

We use the notation $\overline{P} \mid \overline{Q}$ to represent the sequence of processes obtained by putting in parallel the processes of the sequences $\overline{P}$ and $\overline{Q}$ componentwise.

Parallel composition between tagged processes can only be achieved assuming that the shared keys are not revealed. Indeed, if the security of P is ensure through the secrecy of the shared key k, there is no way to guarantee that P is still secure in an environment where another process Q running in parallel will reveal this key.

Since, we consider a common signature Σ_0 and composition contexts with several holes, we have to generalize a bit the notion of revealing a shared key stated in the body of the paper. We have to take into account public keys and verification keys.

Definition 22. *Let C be a composition context, A be an extended process of the form $(\mathcal{E}; C[P_1, \ldots, P_\ell]; \Phi; \sigma)$, and $key \in \{n, \mathsf{pk}(n), \mathsf{vk}(n) \mid n$ occurs in $C\}$. We say that* the extended process A reveals the key key *when:*

- $(\mathcal{E} \cup \{s\}; C[P_1^+, \ldots, P_\ell^+]; \Phi; \sigma) \stackrel{w}{\Rightarrow} (\mathcal{E}'; \mathcal{P}'; \Phi'; \sigma')$ *with* $P_{i_0}^+ \stackrel{\mathsf{def}}{=} P_{i_0} \mid \boldsymbol{in}(c, x).\ \boldsymbol{if}\ x = key\ \boldsymbol{then}\, \boldsymbol{out}(c, s)$ *and* $P_i^+ \stackrel{\mathsf{def}}{=} P_i$ *if* $i \neq i_0$*; and*
- $M\Phi' =_{\mathsf{E}} s$ *for some M such that* $fv(M) \subseteq \mathrm{dom}(\Phi')$ *and* $fn(M) \cap \mathcal{E}' = \emptyset$

where c is a fresh public channel name, s is a fresh name of base type, and the $i_0{}^{th}$ *hole of C is in the scope of "$\boldsymbol{new}\, fn(key)$".*

Definition 23. *Let C be a composition context and $\mathcal{E}_0$ be a finite set of names of base type. Let $\overline{P}$ and $\overline{Q}$ be two sequences of plain processes together with their frames Φ and Ψ. We say that $\overline{P}/\Phi$ and $\overline{Q}/\Psi$ are* composable *under $\mathcal{E}_0$ and C when:*

1. *$\overline{P}$ (resp. $\overline{Q}$) are built over $\Sigma_\alpha \cup \Sigma_0$ (resp. $\Sigma_\beta \cup \Sigma_0$), whereas Φ (resp. Ψ) are built over $\Sigma_\alpha \cup \{\mathsf{pk}, \mathsf{vk}\}$ (resp. $\Sigma_\beta \cup \{\mathsf{pk}, \mathsf{vk}\}$), $\Sigma_\alpha \cap \Sigma_\beta = \emptyset$, and $\overline{P}$ (resp. $\overline{Q}$) is tagged;*
2. *$fv(\overline{P}) = fv(\overline{Q}) = \emptyset$, and $\mathrm{dom}(\Phi) \cap \mathrm{dom}(\Psi) = \emptyset$.*
3. *$\mathcal{E}_0 \cap (fn(C[\overline{P}]) \cup fn(\Phi)) \cap (fn(C[\overline{Q}]) \cup fn(\Psi)) = \emptyset$;*
4. *$(\mathcal{E}_0; C[\overline{P}]; \Phi)$ (resp. $(\mathcal{E}_0; C[\overline{Q}]; \Psi)$) does not reveal any key in:*

$$\{n, \mathsf{pk}(n), \mathsf{vk}(n) \mid n \text{ occurs in } fn(\overline{P}) \cap fn(\overline{Q}) \cap bn(C)\}.$$

This notion is extended as expected to biprocesses requiring that $\mathsf{fst}(\overline{P})/\mathsf{fst}(\Phi)$ and $\mathsf{fst}(\overline{Q})/\mathsf{fst}(\Psi)$, as well as $\mathsf{snd}(\overline{P})/\mathsf{snd}(\Phi)$ and $\mathsf{snd}(\overline{Q})/\mathsf{snd}(\Psi)$, are composable.

Theorem 6. *Let C be a composition context, and $\mathcal{E}_0$ be a finite set of names of base type. Let $\overline{P}$ (resp. $\overline{Q}$) be a sequence of plain biprocesses together with its frame Φ (resp. Ψ), and assume that $\overline{P}/\Phi$ and $\overline{Q}/\Psi$ are composable under $\mathcal{E}_0$ and C.*

If $(\mathcal{E}_0; C[\overline{P}]; \Phi)$ and $(\mathcal{E}_0; C[\overline{Q}]; \Psi)$ satisfy diff-equivalence (resp. trace equivalence), then $(\mathcal{E}_0; C[\overline{P} \mid \overline{Q}]; \Phi \uplus \Psi)$ satisfies diff-equivalence (resp. trace equivalence).

Proof. According to our hypothesis, $\overline{P}$ and $\overline{Q}$ are both tagged hence there exists two sequences of colored plain processes $\overline{P_t}$ and $\overline{Q_t}$ such that $[\![\overline{P_t}]\!] = \overline{P}$ and $[\![\overline{Q_t}]\!] = \overline{Q}$. Moreover, we can split the set of names $\mathcal{E}_0$ into two disjoint sets $\mathcal{E}_P$ and $\mathcal{E}_Q$ depending on whether the name occurs in $\overline{P}/\Phi$ or $\overline{Q}/\Psi$.

Let $S = (\mathcal{E}_0; C[\overline{P} \mid \overline{Q}]; \Phi)$. Our goal is to show that S satisfies diff-equivalence (resp trace equivalence). By hypothesis, we actually have that $(\mathcal{E}_P; C[\overline{P}]; \Phi)$, and $(\mathcal{E}_Q; C[\overline{Q}]; \Psi)$ satisfy diff-equivalence (resp trace equivalence). Let $D = (\mathcal{E}_P \uplus \mathcal{E}_Q; C[\overline{P}] \mid C[\overline{Q}]; \Phi \uplus \Psi)$ (modulo some α-renaming to ensure name and variable distinctness of the resulting process). Since the two processes that are composed in parallel do not share any data, we have that D satisfies diff-equivalence (resp trace equivalence). In order to conclude that S satisfies diff-equivalence (resp trace equivalence), we will show that $\mathsf{fst}(S) \approx_{\mathsf{diff}} \mathsf{fst}(D)$ and $\mathsf{snd}(S) \approx_{\mathsf{diff}} \mathsf{snd}(D)$ relying on Theorem 5.

Let B_1 be the biprocess obtained by forming a biprocess with $\mathsf{fst}(S)$ and $\mathsf{fst}(D)$. Even if the two processes do not have exactly the same structure, this can be achieved by introducing some `new` instructions that will not be used in $\mathsf{fst}(S)$. Relying on Lemma 17, we have that B_1 is in diff-equivalence if and only if $\mathsf{Unf}_n(B_1)$ is in diff-equivalence for any $n \in \mathbb{N}$. Let $n_0 \in \mathbb{N}$. We transform the biprocess $\mathsf{Unf}_{n_0}(B_1)$ to introduce assignment variables (and we may assume w.l.o.g. that the processes under study do not rely on any assignment variables, thus the resulting process will only contain the assignment variables introduced by our transformation), namely $z_1^\alpha, \ldots, z_\ell^\alpha$. This leads us to another biprocess and this transformation still preserves diff-equivalence as stated in Lemma 18. Note that, on the right-hand side of the biprocess (the disjoint case), the assignments variables are assigned to names that do not occur in any action colored β. In order to apply our Theorem 5, we perform a last transformation on this biprocess that consists in replacing the elements that occur inside the frame by output actions (colored with α or β depending on its origin) in front of the biprocess. This last transformation preserves also diff-equivalence. We finally consider $\mathcal{E}_\alpha = \emptyset$, and $\mathcal{E}_\beta = \{k_1^\alpha, \ldots, k_\ell^\alpha\}$ a set of fresh names, and we add these two sets of names to the set of $\mathcal{E}_0$ (first argument of the biprocess). Now, it remains to show that this resulting biprocess B_1' is in diff-equivalence. For this, we rely on Theorem 5. Let ρ_α be such that $\mathrm{dom}(\rho_\alpha) = \emptyset$, and ρ_β be such that $\mathrm{dom}(\rho_\beta) = \{z_1^\alpha, \ldots, z_\ell^\alpha\}$, and $z_j^\alpha \rho_\beta = k_j^\alpha$ for $j \in \{1, \ldots, \ell\}$. Actually, we have that $D_1' = \delta(S_1')$ where $S_1' = \mathsf{fst}(B_1')$ and $D_1' = \mathsf{snd}(B_1')$, and for all possible executions of S_1' or D_1', compatibility will be satisfied. Indeed, by construction, we know that all the assignment variables (remember that all the assignments occurring in the process have been introduced by our transformation) will be assigned to distinct names. Now, to satisfy all the requirements needed to apply Theorem 5, it remains to establish the non-deducibility of the keys.

By hypothesis, $(\mathcal{E}_0; C[\overline{P}]; \Phi)$ and $(\mathcal{E}_0; C[\overline{Q}]; \Psi)$ do not reveal k, $\mathsf{pk}(k)$, or $\mathsf{vk}(k)$ for any $k \in \mathit{fn}(\overline{P}) \cap \mathit{fn}(\overline{Q}) \cap \mathit{bn}(C)$. Hence, we deduce that $\mathsf{fst}(D)$ (parallel composition - disjoint case) does not reveal k, $\mathsf{pk}(k)$, or $\mathsf{vk}(k)$ for any $k \in \mathit{fn}(\overline{P}) \cap \mathit{fn}(\overline{Q}) \cap \mathit{bn}(C)$.

Note that we want to apply Theorem 5 on S_1' and D_1' and not on S_1 and D_1. However, we built D_1' by unfolding D_1 and introducing assignment variables. First, note that these transformations preserve deducibility. Moreover, secrecy of k, $\mathsf{pk}(k)$, or $\mathsf{vk}(k)$ for any $k \in \mathit{fn}(\overline{P}) \cap \mathit{fn}(\overline{Q}) \cap \mathit{bn}(C)$ actually implies that D_1' does not reveal its assignments w.r.t. $(\rho_\alpha, \rho_\beta)$. This allows us to apply Theorem 5 and so to conclude.

F.4 Composing reachability

We now prove a variant of Theorem 1 considering our slightly more general setting.

Theorem 7. *Under the same hypotheses as Theorem 6 with processes instead of bioprocesses, and considering a name s that occurs in C. If $(\mathcal{E}_0; C[\overline{P}]; \Phi)$ and $(\mathcal{E}_0; C[\overline{Q}]; \Psi)$ do not reveal s, then $(\mathcal{E}_0; C[\overline{P} \mid \overline{Q}]; \Phi \cap \Psi)$ does not reveal s.*

Proof. The proof follows the same lines as the one for dealing with diff-equivalence and trace equivalence. In order to show that the process $S = (\mathcal{E}_0; C[\overline{P} \mid \overline{Q}]; \Phi \uplus \Psi)$ does not reveal s, we rely on the fact that the secrecy is preserved by parallel composition of "disjoint" processes. Thanks to our hypotheses, we have that $D = (\mathcal{E}_0; C[\overline{P}] \mid C[\overline{Q}]; \Phi \uplus \Psi)$ does not reveal s. Then, by applying Theorem 5 and more specifically the first bullet point of this theorem, we can deduce that for all $(\mathsf{tr}, \mathtt{new}\ \mathcal{E}_S.\Phi_S) \in \mathsf{trace}(S)$, there exists a trace $(\mathsf{tr}, \mathtt{new}\ \mathcal{E}_D.\Phi_D) \in \mathsf{trace}(D)$ such that $\mathtt{new}\ \mathcal{E}_S.\Phi_S \sim \mathtt{new}\ \mathcal{E}_D.\Phi_D$. Since D does not reveal s, we conclude that S does not reveal s too.

G Sequential composition

In this section we prove Theorems 4 and 2. As for establishing parallel composition results, we will rely on Theorem 5. This will require to unfold the processes under study, and to use assignment variables to share data. However, as already discussed in Section 5, we also have to tackle some additional difficulties. In particular, to ensure the compatibility of the executions as required by Theorem 5.

G.1 Unfolding biprocesses and sharing names via assignments

Unfolding the biprocesses for sequential composition follows the same principles as unfolding the biprocesses for parallel composition. However, we need to be more specific. In particular, we need to be able to easily talk about the replicated instances of a nonce after unfolding. We explain in this section how the unfolded biprocesses are built, and we introduce some notation that we will use throughout the entire section.

Example 16. Let $P = !\mathtt{new}\ k.!\mathtt{new}\ n.\mathtt{out}(c, \mathsf{senc}(n, k))$. The plain process

$$\begin{array}{c}\mathtt{out}(c, \mathsf{senc}(n[1,1], k[1])) \mid \mathtt{out}(c, \mathsf{senc}(n[1,2], k[1])) \\ \mid \mathtt{out}(c, \mathsf{senc}(n[2,1], k[2])) \mid \mathtt{out}(c, \mathsf{senc}(n[2,2], k[2]))\end{array}$$

together with the set

$$\mathcal{K} = \{k[1], k[2], n[1,1], n[1,2], n[2,1], n[2,2]\}$$

will correspond to the 2-unfolding of P, denoted $\mathsf{Unf}_2(P)$. In this example, $k[1], k[2], n[1,1], \ldots, n[2,2]$ are considered as distinct names.

More generally, in such formalism, two names $n_1[i_1,\dots,i_p]$ and $n_2[j_1,\dots,j_q]$ are equal if, and only if, they are syntactically equal, *i.e.* $n_1 = n_2$, $p = q$ and $i_k = j_k$ for each $k \in \{1 \dots p\}$. We will use the same convention to represent the variables occurring in the processes. We will also extend this notation to processes. Thus $P[i_1,\dots,i_n]$ will represent the instance of P that correspond to the i_1^{th} instance of the 1^{st} replication, i_2^{th} instance of the 2^{nd} replication, *etc.*

Example 17. Going back to our previous example, we have that $\mathsf{Unf}_2(P) = (Q[1,1] \mid Q[1,2] \mid Q[2,1] \mid Q[2,2], \mathcal{K})$ where $Q[i,j] = \mathtt{out}(c, \mathsf{senc}(n[i,j], k[i]))$.

With such notation, we can now be much more precise on how our disjoint and shared processes are unfolded.

Following notation given in Theorem 2, we will consider the biprocesses:

1. $S = (\mathcal{E}_0; C[P_1[Q_1] \mid P_2[Q_2]]; \Phi \uplus \Psi)$, the so-called shared case;
2. $D^{\mathsf{par}} = (\mathcal{E}_0; C[P] \mid C[Q]; \Phi \uplus \Psi)$, the so-called parallel disjoint case;
3. $D^{\mathsf{seq}} = (\mathcal{E}_0; \tilde{C}[P_1[\tilde{Q_1}] \mid P_2[\tilde{Q_2}]]; \Phi \uplus \Psi)$ where $\tilde{C}$ is as C but each name n is duplicated $n/\ n^Q$ in order to ensure disjointness. The processes $\tilde{Q_1}$ and $\tilde{Q_2}$ are obtained from Q_1 and Q_2 by replacing each name n occurring in C by its copy n^Q. This represents the so-called sequential disjoint case.

Then, given a biprocess B (typically one given above), we denote by B_n its n^{th} unfolding relying on the naming convention introduced in Example 16 and Example 17.

Using the notation introduced above, it should be clear that for each unfolding n (with $n \in \mathbb{N}$), the biprocess that represents the parallel disjoint case, *i.e.* D_n^{par} exhibits more behaviours than the biprocess that represents the sequential disjoint case, *i.e.* D_n^{seq}.

Lemma 19. *If D_n^{par} satisfies diff-equivalence then D_n^{seq} satisfies diff-equivalence*

As for parallel composition, once unfolding has been done, we get rid of names that are shared through the composition context using assignment variables. We denote these names $r_1, \dots r_p$ and their associated assignment variables $z_1, \dots, z_p$. We also get rid of the content of the frame by adding some outputs in front of the resulting process. Note that, we can assume w.l.o.g. that the only assignment instructions are those that occur in P_1 and P_2 to give a value to x_1 and x_2. Indeed, an assignment of the form $[x := t]$ that is "local" to P_1/P_2 (or Q_1/Q_2) has the same effect as applying the substitution $x \mapsto t$ directly on the process. This additional hypothesis will help us ensure compatibility of all executions when applying Theorem 5.

Given a biprocess B, we will denote B^{v} the biprocess resulting from the transformation described above. In particular, we will consider S_n^{v} the biprocess obtained by applying the transformation above on S_n (the n^{th} unfolding of the shared case), and also D_n^{vseq} the biprocess obtained by applying the transformation on D_n^{seq}.

Again, it should be clear that these transformations preserve diff-equivalence.

Lemma 20. *We have that:*

1. D_n^{vseq} *satisfies diff-equivalence if, and only if,* D_n^{seq} *satisfies diff-equivalence*
2. S_n^{v} *satisfies diff-equivalence if, and only if,* S_n *satisfies diff-equivalence*

Relying on this transformation, by colouring actions of P with α, and actions of Q with β, given an integer n corresponding to the unfolding under study, and assuming that the hole of C is under m replications, we consider ρ_α such that $\mathrm{dom}(\rho_\alpha) = \emptyset$, and ρ_β with

$$\mathrm{dom}(\rho_\beta) = \{z_1, \ldots, z_p\} \quad \cup \{x_1[i_1, \ldots, i_m], x_2[i_1, \ldots, i_m] \mid 1 \leq i_1, \ldots, i_m \leq n\}$$

- $z_i\rho_\beta = r_i$ for $1 \leq i \leq p$;
- $\rho_\beta(x_1[i_1, \ldots, i_m]) = k[i_1, \ldots, i_m]$
- $\rho_\beta(x_2[i_1, \ldots, i_m]) = k[i_1, \ldots, i_m]$.

In other words, we abstract each name shared via the composition context by a fresh one, *i.e.* r_i, and each term shared through the variables x_1 and x_2 are abstracted by a fresh name, a new one for each instance.

G.2 Secrecy of the shared keys

We now focus on the fourth condition of Theorem 5, *i.e.* we ensure that D_n^{vseq} does not reveal the values of its assignments w.r.t. $(\rho_\alpha, \rho_\beta)$ as defined in Section G.1.

Lemma 21. *Assume that* $P_1/P_2/\Phi$ *is a good key-exchange protocol under* $\mathcal{E}_0$ *and* C. *Assume also that* $(\mathcal{E}_0; C[Q]; \Psi; \emptyset)$ *does not reveal any* $k, \mathsf{pk}(k)$ *and* $\mathsf{vk}(k)$.

In such a case, we have that D_n^{vseq} *does not reveal the value of its assignment variables w.r.t.* $(\rho_\alpha, \rho_\beta)$.

Proof. By Definition 7, $P_1/P_2/\Phi$ being a good key-exchange protocol under $\mathcal{E}_0$ and C implies that $(\mathcal{E}_0; P_{\mathsf{good}}; \Phi)$ does not reveal *bad* where P_{good} is defined as follows:

$$\begin{aligned} P_{\mathsf{good}} = {} & \mathtt{new}\, bad.\mathtt{new}\, d.\big(C[\mathtt{new}\, id.(P_1[\mathtt{out}(d, \langle x_1, id\rangle)] \mid P_2[\mathtt{out}(d, \langle x_2, id\rangle)])] \\ & \mid \mathtt{in}(d,x).\mathtt{in}(d,y).\mathtt{if}\ \mathsf{proj}_1(x) = \mathsf{proj}_1(y) \wedge \mathsf{proj}_2(x) \neq \mathsf{proj}_2(y)\ \mathtt{then}\ \mathtt{out}(c, bad) \\ & \mid \mathtt{in}(d,x).\mathtt{in}(d,y).\mathtt{if}\ \mathsf{proj}_1(x) \neq \mathsf{proj}_1(y) \wedge \mathsf{proj}_2(x) = \mathsf{proj}_2(y)\ \mathtt{then}\ \mathtt{out}(c, bad) \\ & \mid \mathtt{in}(d,x).\mathtt{in}(c,z).\mathtt{if}\ z \in \{\mathsf{proj}_1(x), \mathsf{pk}(\mathsf{proj}_1(x)), \mathsf{vk}(\mathsf{proj}_1(x))\}\ \mathtt{then}\ \mathtt{out}(c, bad)\big) \end{aligned}$$

In the case were C is of the form $C'[!\_]$, P_{good} is defined as follows:

$$\begin{aligned} & \mathtt{new}\, bad, d, r_1, r_2.\big(C'[\mathtt{new}\, id.!(P_1[\mathtt{out}(d, \langle x_1, id, r_1\rangle)] \mid P_2[\mathtt{out}(d, \langle x_2, id, r_2\rangle)])] \\ & \mid \mathtt{in}(d,x).\mathtt{in}(d,y).\mathtt{if}\ \mathsf{proj}_1(x) = \mathsf{proj}_1(y) \wedge \mathsf{proj}_2(x) \neq \mathsf{proj}_2(y)\ \mathtt{then}\ \mathtt{out}(c, bad) \\ & \mid \mathtt{in}(d,x).\mathtt{in}(d,y).\mathtt{if}\ \mathsf{proj}_1(x) = \mathsf{proj}_1(y) \wedge \mathsf{proj}_3(x) = \mathsf{proj}_3(y)\ \mathtt{then}\ \mathtt{out}(c, bad) \\ & \mid \mathtt{in}(d,x).\mathtt{in}(c,z).\mathtt{if}\ z \in \{\mathsf{proj}_1(x), \mathsf{pk}(\mathsf{proj}_1(x)), \mathsf{vk}(\mathsf{proj}_1(x))\}\ \mathtt{then}\ \mathtt{out}(c, bad)\big) \end{aligned}$$

In both cases, it indicates that the secrecy of x, $\mathsf{pk}(x)$ and $\mathsf{vk}(x)$ is preserved, where x is the value of any assignment variable. Then, the result is actually a direct consequence of the fact that secrecy is preserved through disjoint composition, and the transformations that are performed on the process (*e.g.* unfolding, adding of some assignments operations) also preserve secrecy.

G.3 Compatibility

To use Theorem 5, a compatibility condition is required. As in the case of parallel composition, this property will be trivially satisfied for assignments that have been added by our transformation. However, more work is needed to deal with assignments present in the original processes, that is in our situation, assignments of the form $[x_1[...] = \_]$ and $[x_2[...] = \_]$ that come from the unfolding of the process P_1/P_2. The idea is that the abstractability property and the fact that $P_1/P_2/\Phi$ is a good key-exchange protocol will give us the required conditions to apply Theorem 5.

The following lemma focuses on $P_1/P_2/\Phi$ being a good key-exchange protocol under $\mathcal{E}_0$ and C. However, the definition of a good key-exchange protocol depends on the shape of the composition context, and the properties satisfied by our processes will depends on the distinction. Hence, to avoid any confusion, unless the composition context is explicitely mentioned being of the form $C'[!\_]$, the definition of good key-exchange protocol always follows Definition 7.

Lemma 22. *Let $(\mathcal{E}_0; C[P_1[0] \mid P_2[0]]; \Phi; \emptyset)$ be a process such that $P_1/P_2/\Phi$ is a good key-exchange protocol under $\mathcal{E}_0$ and C . Let n be an integer, and $(\mathcal{E}; \mathcal{P}; \Phi'; \sigma)$ a process such that $\mathsf{fst}(D_n^{\mathsf{seq}}) \stackrel{\mathsf{tr}}{\Rightarrow} (\mathcal{E}; \mathcal{P}; \Phi'; \sigma)$. Let $i_1, j_1, \ldots, i_m, j_m \in \mathbb{N}$, and $q_1, q_2 \in \{1, 2\}$ such that $x_{q_1}[i_1, \ldots, i_m]$ and $x_{q_2}[j_1, \ldots, j_m]$ are in $\mathrm{dom}(\sigma)$. We have that:*

$$x_{q_1}[i_1, \ldots, i_m]\sigma\downarrow = x_{q_2}[j_1, \ldots, j_m]\sigma\downarrow$$
$$\text{if, and only if,}$$
$$i_p = j_p \text{ for all } 1 \leq p \leq m.$$

A similar property holds for $\mathsf{snd}(D_n^{\mathsf{seq}})$.

Proof. By definition of $P_1/P_2/\Phi$ being a good key-exchange protocol under $\mathcal{E}_0$ and C and since secrecy is preserved when considering disjoint composition, we have that $(\mathcal{E}_0; P; \Phi \uplus \Psi; \emptyset)$ preserves the secrecy of *bad* where:

$$\begin{array}{l}
P = \mathtt{new}\, bad.\, \mathtt{new}\, d.(\\
\quad \tilde{C}[\mathtt{new}\, k.\mathtt{new}\, id.(P_1[\mathtt{out}(d, \langle x_1, id\rangle).\tilde{Q}_1\{^k/_{x_1}\}] \\
\qquad\qquad\qquad\quad \mid P_2[\mathtt{out}(d, \langle x_2, id\rangle).\tilde{Q}_2\{^k/_{x_2}\}])] \\
\quad \mid \mathtt{in}(d, x).\mathtt{in}(d, y). \\
\qquad \mathtt{if}\ \mathsf{proj}_1(x) = \mathsf{proj}_1(y) \wedge \mathsf{proj}_2(x) \neq \mathsf{proj}_2(y) \\
\qquad \mathtt{then}\, \mathtt{out}(c, bad) \\
\qquad \mathtt{else}\, \mathtt{if}\ \mathsf{proj}_1(x) \neq \mathsf{proj}_1(y) \wedge \mathsf{proj}_2(x) = \mathsf{proj}_2(y) \\
\qquad \mathtt{then}\, \mathtt{out}(c, bad) \\
\quad)
\end{array}$$

Here, the notation $\tilde{C}$, $\tilde{Q}_1$, and $\tilde{Q}_2$ refer to the same renaming as the one used to define D^{seq}.

Let n be an integer. Consider the n^{th} unfolding of D^{seq} as well as the n^{th} unfolding of the process P defined above. First, note that an output on channel d is always of the form

$$\mathtt{out}(d, \langle x_j[i_1, \dots i_m], id[i_1, \dots i_m]\rangle) \text{ with } j \in \{1, 2\}.$$

Let $(\mathcal{E}; \mathcal{P}; \Phi'; \sigma)$ be a process such that $\mathsf{fst}(D_n^{\mathsf{seq}}) \overset{\mathsf{tr}}{\Rightarrow} (\mathcal{E}; \mathcal{P}; \Phi'; \sigma)$ with $x_{q_1}[i_1, \dots, i_m]$ and $x_{q_2}[j_1, \dots, j_m]$ both in $\mathrm{dom}(\sigma)$. Moreover, assume that $x_{q_1}[i_1, \dots, i_m]\sigma\!\downarrow = x_{q_2}[j_1, \dots, j_m]\sigma\!\downarrow$. In such a case, it is easy to build a trace of $(\mathcal{E}_0; P_n; \Phi \uplus \Psi; \emptyset)$ such that the pairs

- $\langle x_{q_1}[i_1, \dots, i_m], id[i_1, \dots, i_m]\rangle$, and
- $\langle x_{q_2}[j_1, \dots, j_m], id[j_1, \dots, j_m]\rangle$

are outputted on channel d. Since the hole in P_{q_1} (resp. P_{q_2}) is not in the scope of a replication, we deduce that these pairs can only be outputted once. We have seen that such a process preserves the secrecy of bad, and thus we deduce that $(i_1, \dots, i_p) = (j_1, \dots, j_p)$.

Now, relying on the fact that $(\mathcal{E}_0; P_n; \Phi \uplus \Psi; \emptyset)$ preserves the secrecy of bad, and more precisely on the fact that the following instructions are part of the process:

$$\begin{array}{l} \mid \mathtt{in}(d, x).\mathtt{in}(d, y). \\ \quad \dots \\ \quad \mathtt{else\,if}\ \mathsf{proj}_1(x) \neq \mathsf{proj}_1(y) \wedge \mathsf{proj}_2(x) = \mathsf{proj}_2(y) \\ \quad \mathtt{then\,out}(c, bad) \end{array}$$

we deduce that $(i_1, \dots, i_p) = (j_1, \dots, j_p)$ implies that $id[i_1, \dots, i_m] = id[j_1, \dots, j_m]$ and so we deduce that $x_{q_1}[i_1, \dots, i_m]\sigma\!\downarrow = x_{q_2}[j_1, \dots, j_m]\sigma\!\downarrow$.

Note that the property above we established for D_n^{seq} also holds on D_n^{vseq}. We have also a similar result in case the composition context is of the form $C'[!\_]$ that is stated below and can be proved in a similar way.

Lemma 23. *Let $(\mathcal{E}_0; C[P_1[0] \mid P_2[0]]; \Phi; \emptyset)$ be a process such that $C = C'[!\_]$ for some C' and $P_1/P_2/\Phi$ is a good key-exchange protocol under $\mathcal{E}_0$ and C. Let n be an integer and $(\mathcal{E}; \mathcal{P}; \Phi'; \sigma)$ be a process such that $\mathsf{fst}(D_n^{\mathsf{seq}}) \overset{\mathsf{tr}}{\Rightarrow} (\mathcal{E}; \mathcal{P}; \Phi'; \sigma)$.*
Let $i_1, j_1, \dots, i_m, j_m \in \mathbb{N}$. We have that:

- *$x_1[i_1, \dots, i_m]\sigma\!\downarrow = x_2[j_1, \dots, j_m]\sigma\!\downarrow$ implies that $(i_1, \dots, i_{m-1}) = (j_1, \dots, j_{m-1})$; and*
- *for $q \in \{1, 2\}$, $x_q[i_1, \dots, i_m]\sigma\!\downarrow = x_q[j_1, \dots, j_m]\sigma\!\downarrow$ implies $(i_1, \dots, i_m) = (j_1, \dots, j_m)$.*

A similar property holds for $\mathsf{snd}(D_n^{\mathsf{seq}})$.

Now, regarding assingment variables, and in particular the different instances of x_1 and x_2, it remains to show that the values assigned to these variables will be rooted in the right signature. We proceed in two steps. First, we discard terms rooted with a symbol in $\{\mathsf{pk}, \mathsf{vk}, \langle\rangle\}$ (Lemma 24), and then we show that it is actually rooted in the right signature (Lemma 25).

Definition 24. *We say that a process P satisfies the* abstractability property *if for all $P \stackrel{\mathsf{tr}}{\Rightarrow} (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$, for all assignment variable $x \in \mathrm{dom}(\sigma)$, $\mathsf{root}(x\sigma\!\downarrow) \notin \{\mathsf{pk}, \mathsf{vk}, \langle\rangle\}$.*

This property is important for our composition to hold.

Example 18. Let $P_i = [x_i := \langle k_1, k_2 \rangle]$, and $Q_i = \mathtt{if}\, x_i = \langle \mathsf{proj}_1(x_i), \mathsf{proj}_2(x_i) \rangle$ $\mathtt{then}\, \mathtt{out}(c, id_i)$. Let $C = \mathtt{new}\, k_1.\mathtt{new}\, k_2.\_$. We can see that in the shared case, the branch Then of the process Q_i will be executed whereas when considering in isolation the process $C[\mathtt{new}\, k.(Q_1\{x_1 \mapsto k\} \mid Q_2\{x_2 \mapsto k\})]$ will not exhibit a similar behaviour.

Intuitively, we say that a value of an assignment variable can be abstracted if it is not a pair, a public key or verification key. This is due to the fact that those three primitives are not tagged and so can be used by processes of any colour.

Lemma 24. *Let $(\mathcal{E}_0; C[P_1[0] \mid P_2[0]]; \Phi; \emptyset)$ be a process satisfying the abstractability property. We have that D^{vseq} satisfies the abstractability property.*

Proof. First of all, unfolding the process $(\mathcal{E}_0; C[P_1[0] \mid P_2[0]]; \Phi; \emptyset)$ preserves the abstractability property. Moreover, the transformation that transforms a process B into a process B^{v} preserves the abstractability property. Thus, to show that D^{vseq} satisfies the abstractability property, it only remains to show that this property is preserved by disjoint composition assuming that the process we want to compose does not introduce new assignments (note that this is the case of Q_1/Q_2).

In fact, part of the process brought by Q_1/Q_2 can be viewed as a process executed by the attacker. Thus, for all $D^{\mathsf{seq}} \stackrel{\mathsf{tr}'}{\Longrightarrow} (\mathcal{E}'; \mathcal{P}'; \Phi'; \sigma')$, there exists a correspondig execution $(\mathcal{E}_0; C[P_1[0] \mid P_2[0]]; \Phi; \emptyset) \stackrel{\mathsf{tr}''}{\Longrightarrow} (\mathcal{E}''; \mathcal{P}''; \Phi''; \sigma'')$ such that σ'' and σ' coincide on $\mathrm{dom}(\sigma'')$, and in particular on the values assigned to $x_1[\ldots]$ and $x_2[\ldots]$. This allows us to deduce that $\mathsf{root}(x\sigma'') \notin \{\mathsf{pk}, \mathsf{vk}, \langle\rangle\}$, and thus D^{vseq} satisfies the abstractability property.

The next lemma will allow us to conclude that we obtain traces compatible with $(\rho_\alpha, \rho_\beta)$.

Lemma 25. *Assume that D_n^{vseq} does not reveal the value of its assignment variables w.r.t. $(\rho_\alpha, \rho_\beta)$ and satisfies the abstractability property. We have that for all $D_n^{\mathsf{vseq}} \stackrel{\mathsf{tr}}{\Rightarrow} (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$, for all $\gamma \in \{\alpha, \beta\}$, for all $z \in \mathrm{dom}(\rho_\gamma)$, we have that either $\mathsf{tagroot}(z\sigma\!\downarrow) = \perp$ or $\mathsf{tagroot}(z\sigma\!\downarrow) \notin \gamma \cup \{0\}$.*

Proof. Since $D_n^{\mathsf{vseq}} \stackrel{\mathsf{tr}}{\Rightarrow} (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$, we know that $(\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ is a derived well-tagged extended process w.r.t. $\prec$ and *col*, for some $\prec$ and *col*. Moreover, by construction of D_n^{vseq}, we also know that $\mathrm{dom}(\rho_\alpha) = \emptyset$. We prove the result by induction of the $\mathrm{dom}(\rho_\beta)$ with the order $\prec$.

Base case $z \prec z'$ for all assignment variables z' different from z: Assume that $\mathsf{tagroot}(z\sigma\downarrow) \neq \bot$ and $\mathsf{tagroot}(z\sigma\downarrow) \in \beta \cup \{0\}$. We now show that $z\sigma\downarrow \in Fct_\alpha(z\sigma)$. Since $\mathsf{tagroot}(z\sigma\downarrow) \in \beta \cup \{0\}$, we have that if $\mathsf{root}(z\sigma\downarrow) \not\in \{\mathsf{vk}, \mathsf{pk}, \langle\ \rangle\}$ then $z\sigma \in Fct_\alpha(z\sigma)$. Thus it remains to show that $\mathsf{root}(z\sigma\downarrow) \not\in \{\mathsf{vk}, \mathsf{pk}, \langle\ \rangle\}$. But D_n^{vseq} satisfies the abstractability property hence we deduce that $\mathsf{root}(z\sigma\downarrow) \not\in \{\mathsf{vk}, \mathsf{pk}, \langle\ \rangle\}$.

Since $z\sigma\downarrow \in Fct_\alpha(z\sigma)$, we can apply Lemma 12 and so we deduce that:

1. either there exists M such that $fv(M) \subseteq \mathrm{dom}(\Phi) \cap \{z' \mid z' \prec z\}$, $fn(M) \cap \mathcal{E} = \emptyset$ and $z\sigma\downarrow \in Fct_\gamma(M\Phi\downarrow)$
2. otherwise there exists j such that $z_j^\beta \prec z$ and $z_j^\beta\sigma\downarrow = z\sigma\downarrow$

The second case is trivially impossible since $\mathrm{dom}(\rho_\alpha) = \emptyset$ and so z_j^β does not exists. We focus on the first case: We know that $z\sigma\downarrow \in Fct_\gamma(M\Phi\downarrow)$. Since $z\sigma\downarrow$ is not deducible in $\mathtt{new}\ \mathcal{E}.\Phi$, then $z\sigma\downarrow \not\in Fct_{\langle\ \rangle}(M\Phi\downarrow)$. Moreover, we know that for all assignment variables z' different from z, $z \prec z'$. Thus we can apply Lemma 14 and obtain that there exists M'' such that $fv(M) \subseteq \mathrm{dom}(\Phi)$, $fn(M) \cap \mathcal{E} = \emptyset$ and $z\sigma\downarrow \in Fct_{\langle\ \rangle}(M'\Phi\downarrow)$. But this contradicts the fact that $z\sigma\downarrow$ is not deducible in $\mathtt{new}\ \mathcal{E}.\Phi$.

Since we always reach a contradiction, we can conclude that $\mathsf{tagroot}(z\sigma\downarrow) = \bot$ or $\mathsf{tagroot}(z\sigma\downarrow) \not\in \beta \cup \{0\}$.

Inductive case: Assume once again that $\mathsf{tagroot}(z\sigma\downarrow) \neq \bot$ and $\mathsf{tagroot}(z\sigma\downarrow) \in \beta \cup \{0\}$. As in the previous case, we can show that $z\sigma\downarrow \in Fct_\alpha(z\sigma)$ and so we can apply Lemma 12 to obtain:

1. either there exists M such that $fv(M) \subseteq \mathrm{dom}(\Phi) \cap \{z' \mid z' \prec z\}$, $fn(M) \cap \mathcal{E} = \emptyset$ and $z\sigma\downarrow \in Fct_\gamma(M\Phi\downarrow)$
2. otherwise there exists j such that $z_j^\beta \prec z$ and $z_j^\beta\sigma\downarrow = z\sigma\downarrow$

Once again the first case is trivially impossible since $\mathrm{dom}(\rho_\alpha) = \emptyset$. Thus it remain to focus on the second case. As in the previous, we can deduce that $z\sigma\downarrow \not\in Fct_{\langle\ \rangle}(M\Phi\downarrow)$. Moreover, by our inductive hypothesis, we know that for all assignment variable $z' \prec z$, $\mathsf{tagroot}(z'\sigma\downarrow) = \bot$ or $\mathsf{tagroot}(z'\sigma\downarrow) \not\in \beta \cup \{0\}$. Thus, we can deduce that $z'\sigma\downarrow \neq z\sigma\downarrow$. Thanks to this, we can apply Lemma 14 and obtain that there exists M'' such that $fv(M) \subseteq \mathrm{dom}(\Phi)$, $fn(M) \cap \mathcal{E} = \emptyset$ and $z\sigma\downarrow \in Fct_{\langle\ \rangle}(M'\Phi\downarrow)$. But this contradicts the fact that $z\sigma\downarrow$ is not deducible in $\mathtt{new}\ \mathcal{E}.\Phi$.

Since we always reach a contradiction, we can conclude that $\mathsf{tagroot}(z\sigma\downarrow) = \bot$ or $\mathsf{tagroot}(z\sigma\downarrow) \not\in \beta \cup \{0\}$.

We now establish that when the processes are a good key exchanged protocol, all possible executions are actually compatible w.r.t. $(\rho_\alpha, \rho_\beta)$.

Lemma 26. *Let $(\rho_\alpha, \rho_\beta)$ be the two abstraction functions as defined in Section G.1. If $(\mathcal{E}_0; C[P_1[0] \mid P_2[0]]; \Phi; \emptyset)$ satisfies the abstractability property and $P_1/P_2/\Phi$ is a good key-exchanged protocol under $\mathcal{E}_0$ and C then for any P such that:*

– $\mathsf{fst}(D_n^{\mathsf{vseq}}) \stackrel{\mathsf{tr}}{\Rightarrow} P$ *(resp.* $\mathsf{snd}(D_n^{\mathsf{vseq}}) \stackrel{\mathsf{tr}}{\Rightarrow} P$*), we have that* P *is compatible w.r.t.* $(\rho_\alpha, \rho_\beta)$.
– $\mathsf{fst}(S_n^{\mathsf{v}}) \stackrel{\mathsf{tr}}{\Rightarrow} P$ *(resp.* $\mathsf{snd}(S_n^{\mathsf{v}}) \stackrel{\mathsf{tr}}{\Rightarrow} P$*), we have that* P *is compatible w.r.t.* $(\rho_\alpha, \rho_\beta)$.

Proof. Let P be a process such that $\mathsf{fst}(D_n^{\mathsf{vseq}}) \stackrel{\mathsf{tr}}{\Rightarrow} (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$. Let $x, y \in \mathrm{dom}(\sigma) \cap \mathrm{dom}(\rho_\beta)$ and assume that $x\sigma =_{\mathsf{E}} y\sigma$. Let us denote $x = x_i[i_1, \dots, i_m]$ and $y = x_j[j_1, \dots, j_m]$ where $j_k, i_k \in \{1, \dots, n\}$, $k \in \{1, \dots, m\}$ and $i, j \in \{1, 2\}$.

By hypothesis, $P_1/P_2/\Phi$ is a good key-exchanged protocol under $\mathcal{E}_0$ and C . Hence thanks to Lemma 22, $x\sigma =_{\mathsf{E}} y\sigma$ implies that $i_k = j_k$ for all $k \in \{1 \dots m\}$. On the other hand, Lemma 22 also indicates that $x_1[i_1, \dots, i_m]\sigma = x_2[i_1, \dots, i_m]\sigma$, for all $i_1, \dots, i_m$.

Since by definition of ρ_β, $x_1[i_1, \dots, i_m]\rho_\beta = x_2[i_1, \dots, i_m]\rho_\beta = k[i_1, \dots, i_m]$, we can deduce that $x\sigma = y\sigma$ if and only if $x\rho_\beta = y\rho_\beta$. At last, relying on Lemma 25, we can conclude that $(\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ is compatible with $(\rho_\alpha, \rho_\beta)$.

We now prove the property for S_n^{v}: Let $\mathsf{fst}(S_n^{\mathsf{v}}) \stackrel{\mathsf{tr}}{\Rightarrow} (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$. We prove the result by induction on the size of tr. Consider a transition $(\mathcal{E}; \mathcal{P}; \Phi; \sigma) \xrightarrow{\ell} A$. By inductive hypothesis, we know that $(\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ is compatible with $(\rho_\alpha, \rho_\beta)$. But, the only transition that could render A not compatible is the internal transition (Assgn). Hence assume that $\mathcal{P} = \{[x := t]^i.P\} \uplus \mathcal{Q}$ where $i \in \gamma$ and $\ell = \tau$.

Since $(\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ is compatible, then by Theorem 5 and in particular Proposition 1, we deduce that $\mathsf{fst}(D_n^{\mathsf{vseq}}) \stackrel{\mathsf{tr}}{\Rightarrow} (\mathcal{E}'; \mathcal{P}'; \Phi'; \sigma')$ where $\delta(\sigma\!\downarrow) = \sigma'\!\downarrow$ and $\delta(\mathcal{P}) = \mathcal{P}'$. It implies that $\mathcal{P}' = \{[x := \delta_\gamma(t)]^i.\delta(P)\} \uplus \delta(\mathcal{Q})$. Thus, by Lemma 7, we have that $\delta_\gamma(t\sigma\!\downarrow) = \delta_\gamma(t)\sigma'\!\downarrow$.

On the other hand, if A is not compatible, it means that there exists $y \in \mathrm{dom}(\sigma)$ such that $t\sigma\!\downarrow = y\sigma\!\downarrow$ is not equivalent to $x\rho_\beta = y\rho_\beta$. But $\delta(\sigma\!\downarrow) = \sigma'\!\downarrow$ and $\delta_\gamma(t\sigma\!\downarrow) = \delta_\gamma(t)\sigma'\!\downarrow$. Hence $t\sigma\!\downarrow = y\sigma\!\downarrow$ is equivalent to $\delta_\gamma(t)\sigma' = y\sigma'$, and so we can deduce that $\delta_\gamma(t)\sigma' = y\sigma'$ is not equivalent to $x\rho_\beta = y\rho_\beta$. However, $(\mathcal{E}'; \mathcal{P}'; \Phi'; \sigma')$ can also apply the internal transition (Assgn) on $[x := \delta(t)]^i$ and so we obtain $(\mathcal{E}'; \mathcal{P}'; \Phi'; \sigma') \xrightarrow{\tau} A'$ with A' not compatible with $(\rho_\alpha, \rho_\beta)$. This is in contradiction with our result on D_n^{vseq}.

When the composition context is of the form $C'[!\_]$, the previous lemma does not hold. However, we will show that we can modify any trace to become a compatible trace by applying some permutation on the indices of the names. Intuitively, when considering a trace of S_n^{v}, if $x_1[i_1, \dots, i_m]$ is equal to $x_2[i_1, \dots, i_{m-1}, i'_m]$ after instantiation with $i_m \neq i'_m$, we want to permute all names of the form $t[i_1, \dots, i_{m-1}, i'_m]$ by $t[i_1, \dots, i_{m-1}, i_m]$. Such permutation is possible since we only consider composition context of the form $C'[!\_]$. We will call this an *index permutation.* To ensure that such a permutation is always possible when needed, we simply ensure that we have enough processes that have not started their execution by requiring that $2n' \leq n$ (*i.e.* the length n' of the derivation under study is two times smaller than the number of the unfolding we consider).

Lemma 27. *Let $(\rho_\alpha, \rho_\beta)$ be the two abstraction functions of S_n^{v}. For all $S_n^{\mathsf{v}} \xrightarrow{\ell_1} A_1 \xrightarrow{\ell_2} \dots \xrightarrow{\ell_{n'}} A_{n'}$ with $2n' < n$, there exists an index permutation such that $S_n^{\mathsf{v}} \xrightarrow{\ell_1} A'_1 \xrightarrow{\ell_2} \dots \xrightarrow{\ell_{n'}} A'_{n'}$ with A'_k being the application of the index permutation on A_k for all $k = 1 \dots n'$, and $A'_{n'}$ is compatible with $(\rho_\alpha, \rho_\beta)$.*

Proof. We prove the result by induction on n'. The initial step $n' = 0$ being trivial, we focus on the inductive step $n' > 0$. By hypothesis, we know that there exists an index permutation such that $S_n^{\mathsf{v}} \xrightarrow{\ell_1} A'_1 \xrightarrow{\ell_2} \dots \xrightarrow{\ell_{n'}} A'_{n'-1}$ where A'_k being the application of the index permutation on A_k for all $k = 1 \dots n'-1$, and $A'_{n'-1}$ is compatible with $(\rho_\alpha, \rho_\beta)$. However, we know that $A_{n'-1} \xrightarrow{\ell_{n'}} A_{n'}$. Since $A'_{n'-1}$ is obtained from $A_{n'-1}$ by an index permutation, then $A'_{n'-1} \xrightarrow{\ell_{n'}} A'_{n'}$ where $A'_{n'}$ is the application of the index permutation on $A_{n'}$.

Assume first that the transition $A_{n'-1} \xrightarrow{\ell_{n'}} A_{n'}$ is different from the internal transition (Assgn), then the compatibility of $A'_{n'-1}$ implies the compatibility of $A'_{n'}$. Hence the result holds.

Assume now that the transition $A_{n'-1} \xrightarrow{\ell_{n'}} A_{n'}$ is the internal transition (Assgn). Consider that $A'_{n'-1} = (\mathcal{E}; \mathcal{P}; \Phi; \sigma)$ with $\mathcal{P} = \{[x := t]^i.P\} \uplus \mathcal{Q}$. Since $A'_{n'-1}$ is compatible with $(\rho_\alpha, \rho_\beta)$, we can apply Proposition 1. Using similar reasoning as in proof of Lemma 26, we obtain that $t\sigma = y\sigma$ for some assignment variable y implies w.l.o.g. that $x = x_1[i_1, \dots, i_{m-1}, i_m]$ and $y = x_2[i_1, \dots, i_{m-1}, i'_m]$. Thus, by applying the index permutation between i_m and i'_m on each A'_k, we obtain that $S_n^v \xrightarrow{\ell_1} A''_1 \xrightarrow{\ell_2} \dots \xrightarrow{\ell_{n'}} A''_{n'}$ with $A''_{n'}$ compatible with $(\rho_\alpha, \rho_\beta)$, and A''_k being the application of the index permutation on A'_k, for all $k \in \{1, \dots, m\}$.

G.4 Composing diff-equivalence

We are now able to prove our composition results.

*

Proof. Let $S = (\mathcal{E}_0; C[P_1[Q_1] \mid P_2[Q_2]]; \Phi \uplus \Psi; \emptyset)$. Thanks to Lemma 17, we know that S is in diff-equivalence if, and only if, S_n is in diff-equivalence for all $n \in \mathbb{N}$.

By hypothesis, we know that:

- $(\mathcal{E}_0; C[P_1[0] \mid P_2[0]]; \Phi; \emptyset)$, and
- $(\mathcal{E}_0; C[\mathtt{new}\ k.(Q_1\{^k/_{x_1}\} \mid Q_2\{^k/_{x_2}\})]; \Psi; \emptyset)$

are both in diff-equivalence and P_1, P_2, Q_1, Q_2 are tagged. Hence, since diff-equivalence is preserved by disjoint parallel composition, we deduce that D^{par} is in diff-equivalence, and thus, thanks to Lemma 17, we obtain that D_n^{par} is in diff-equivalence for all $n \in \mathbb{N}$. Applying Lemma 19, we deduce that D_n^{seq} is also in diff-equivalence. Note that diff-equivalence still holds on the biprocess D_n^{vseq} obtained from D_n^{seq} by adding some assignment variables to "explicit the sharing".

Given $n \in \mathbb{N}$, in order to conclude, we have to show that S_n^{v} obtained from S_n by adding some assignments variables to explicit the sharing satisfies diff-equivalence. We form two new biprocesses SD_L and SD_R as follows:

- $\mathsf{fst}(SD_L) = \mathsf{fst}(S_n^{\mathsf{v}})$ and $\mathsf{snd}(SD_L) = \mathsf{fst}(D_n^{\mathsf{vseq}})$;
- $\mathsf{fst}(SD_R) = \mathsf{snd}(S_n^{\mathsf{v}})$ and $\mathsf{snd}(SD_R) = \mathsf{snd}(D_n^{\mathsf{vseq}})$;

We will apply Theorem 5 on biprocesses SD_L and SD_R to establish the strong relationship between the two components of each biprocess, and together with the fact D_n^{vseq} satisfies diff-equivalence, this will allow us to conclude that S_n^{v} satisfies diff-equivalence too.

Considering the two abstraction functions $(\rho_\alpha, \rho_\beta)$ as defined in Section G.1, in order to apply Theorem 5 on SD_L (resp. SD_R), it remains to show that $\mathsf{fst}(D_n^{\mathsf{vseq}})$ and $\mathsf{snd}(D_n^{\mathsf{vseq}})$ do not reveal the value of their assignment variables w.r.t. $(\rho_\alpha, \rho_\beta)$. This is actually achieved by application of Lemma 21 with the facts that

- $(\mathcal{E}_0; C[\mathtt{new}\ k.(Q_1\{^k/_{x_1}\} \mid Q_2\{^k/_{x_2}\})]; \Psi; \emptyset)$ and $(\mathcal{E}_0; C[P_1[0] \mid P_2[0]]; \Phi; \emptyset)$ do not reveal key in $\{n, \mathsf{pk}(n), \mathsf{vk}(n) \mid n \in fn(P_1, P_2) \cap fn(Q_1, Q_2) \cap bn(C)\}$, and
- $(\mathcal{E}_0; C[\mathtt{new}\ k.(Q_1\{^k/_{x_1}\} \mid Q_2\{^k/_{x_2}\})]; \Psi; \emptyset)$ do not reveal $k, \mathsf{pk}(k), \mathsf{vk}(k)$, and
- $P_1/P_2/\Phi$ is a good key-exchange protocol under $\mathcal{E}_0$ and C, that implies in particular that $(\mathcal{E}_0; P_{\mathsf{good}}; \Phi)$ does not reveal *bad* where P_{good} is defined as follows:

$$\begin{array}{l} P_{\mathsf{good}} = \mathtt{new}\, bad, d.\big(\\ \quad C[\mathtt{new}\, id.(P_1[\mathtt{out}(d, \langle x_1, id\rangle)] \mid P_2[\mathtt{out}(d, \langle x_2, id\rangle)])] \\ \quad \mid \mathtt{in}(d, x).\mathtt{in}(c, z). \\ \qquad \mathtt{if}\, z \in \{\mathsf{proj}_1(x), \mathsf{pk}(\mathsf{proj}_1(x)), \mathsf{vk}(\mathsf{proj}_1(x))\} \\ \qquad \mathtt{then}\, \mathtt{out}(c, bad)\big) \end{array}$$

Now, let B_S be a biprocess such that

$$S_n^{\mathsf{v}} \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} B_S \overset{\mathsf{def}}{=} (\mathcal{E}_S; \mathcal{P}_S; \Phi_S; \sigma_S)$$

for some tr. By definition of diff-equivalene, we have to show that:

1. $\mathtt{new}\ \mathcal{E}_S.\mathsf{fst}(\Phi_S) \sim \mathtt{new}\ \mathcal{E}_S.\mathsf{snd}(\Phi_S)$;
2. if $\mathsf{fst}(B_S) \xrightarrow{\ell} A_L$ then there exists B' such that $B_S \xrightarrow{\ell}_{\mathsf{bi}} B'$ and $\mathsf{fst}(B') = A_L$ (and similarly for snd).

Let us now focus on the case where the composition context is not of the form $C[!\_]$.

We have $\mathsf{fst}(S_n^{\mathsf{v}}) \overset{\mathsf{tr}}{\Rightarrow} \mathsf{fst}(B_S)$ as well as $\mathsf{snd}(S_n^{\mathsf{v}}) \overset{\mathsf{tr}}{\Rightarrow} \mathsf{snd}(B_S)$. By Lemma 26, we obtain that $\mathsf{fst}(B_S)$ as well as $\mathsf{snd}(B_S)$ is compatible with $(\rho_\alpha, \rho_\beta)$. Hence, relying on Theorem 5 (first item), we deduce that there exist biprocesses SD'_L and SD'_R such that:

- $SD_L \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} SD'_L$, $\mathsf{fst}(SD'_L) = \mathsf{fst}(B_S)$, and static equivalence holds between the two frames issued from the biprocess SD'_L;

– $SD_R \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} SD'_R$, $\mathsf{fst}(SD'_R) = \mathsf{snd}(B_S)$, and static equivalence holds between the two frames issued from the biprocess SD'_R.

Since, we know that D_n^{vseq} satisfies diff-equivalence, we have that $D_n^{\mathsf{vseq}} \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} D_n'^{\mathsf{vseq}}$ with $\mathsf{fst}(D_n'^{\mathsf{vseq}}) = \mathsf{snd}(SD'_L)$ and $\mathsf{snd}(D_n'^{\mathsf{vseq}}) = \mathsf{snd}(SD'_R)$. Then, by transitivity of static equivalence, we deduce that

$$\mathtt{new}\ \mathcal{E}_S.\mathsf{fst}(\Phi_S) \sim \mathtt{new}\ \mathcal{E}_S.\mathsf{snd}(\Phi_S).$$

Now, assume that $\mathsf{fst}(B_S) \xrightarrow{\ell} A_L$. In such a case, we have that $\mathsf{fst}(S_n) \overset{\mathsf{tr}}{\Rightarrow} \mathsf{fst}(B_S) \xrightarrow{\ell} A_L$. By Lemma 26, we obtain that A_L is compatible with $(\rho_\alpha, \rho_\beta)$, and relying on Theorem 5 (first item), we deduce that there exists a biprocess SD''_L such that: $SD_L \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} \_ \xrightarrow{\ell}_{\mathsf{bi}} SD''_L$ with $\mathsf{fst}(SD''_L) = A_L$. Since D_n^{vseq} satisfies diff-equivalence, we have that $D_n^{\mathsf{vseq}} \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} \xrightarrow{\ell}_{\mathsf{bi}} D_n''^{\mathsf{vseq}}$ for some biprocess $D_n''^{\mathsf{vseq}}$ with $\mathsf{fst}(D_n''^{\mathsf{vseq}}) = \mathsf{snd}(SD''_L)$. Now, applying Theorem 5 (second item) on biprocess SD_R, we deduce that $SD_R \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} \_ \xrightarrow{\ell}_{\mathsf{bi}} SD''_R$ with $\mathsf{snd}(SD''_R) = \mathsf{snd}(D_n''^{\mathsf{vseq}})$. This allows us to ensure the existence of the biprocess B' required to show diff-equivalence of S_n^{v}. We will have $\mathsf{fst}(B') = \mathsf{fst}(SD''_L) = A_L$ and $\mathsf{snd}(B') = \mathsf{fst}(SD''_R)$.

In the case where the composition context is of the form $C'[!\_]$, all the traces issued from S_n^{v} are not compatible anymore w.r.t. the abstraction functions ρ_α and ρ_β. Nevertheless, thanks to Lemma 27, we can always find a similar trace that is compatible, then using Theorem 5, we will ensure that these traces also exist in the disjoint case, and we also ensure their compatibility (see Proposition 1).

Then, relying on the diff-equvialence of the biprocess $(\mathcal{E}_0; \mathtt{new}\, d.C[P^+]; \Phi)$, we deduce that for any trace $D_n^{\mathsf{vseq}} \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} D'$, $\mathsf{fst}(D')$ is compatible w.r.t. $(\rho_\alpha, \rho_\beta)$ if and only if $\mathsf{snd}(D')$ is compatible w.r.t. $(\rho_\alpha, \rho_\beta)$. This allows us to ensure that D_n^{seq} is also in diff-equivalence when considering compatible traces only. Thanks to this, we are able to conclude as in we did in the case where the composition context were not of the form $C'[!\_]$.

G.5 Composing reachability

*

Proof. Let $S = (\mathcal{E}_0; C[P_1[Q_1] \mid P_2[Q_2]]; \Phi \uplus \Psi; \emptyset)$. By hypothesis, we know that:

– $(\mathcal{E}_0; C[P_1[0] \mid P_2[0]]; \Phi; \emptyset)$, and
– $(\mathcal{E}_0; C[\mathtt{new}\ k.(Q_1\{^k/_{x_1}\} \mid Q_2\{^k/_{x_2}\})]; \Psi; \emptyset)$

does not reveal s. Since, secrecy is preserved by disjoint composition, and the transformations introduced at the beginning of the section (*e.g.* unfolding, adding assignment variables, ...), we easily deduce that D_n^{vseq} do not reveal s.

We show the result by contradiction. Assume that S_n^{v} reveals the secrecy s. We consider a trace witnessing this fact, *i.e.* a process $S_n'^{\mathsf{v}}$ such that

$$S_n^{\mathsf{v}} \overset{\mathsf{tr}}{\Rightarrow} S_n'^{\mathsf{v}} \overset{\mathsf{def}}{=} (\mathcal{E}_S; \mathcal{P}_S; \Phi_S; \sigma_S)$$

and for which $\mathtt{new}\ \mathcal{E}_S.\Phi_S \vdash s$.

We form a biprocess SD by grouping together S_n^{v} and D_n^{vseq} in order to apply Theorem 5.

In order to apply Theorem 5, we first must prove that D_n^{vseq} does not reveal the value of its assignment variables w.r.t. $(\rho_\alpha, \rho_\beta)$ as defined in Section G.1. This is achieved by application of Lemma 21 with the facts that

– $(\mathcal{E}_0; C[\mathtt{new}\ k.(Q_1\{^k/_{x_1}\} \mid Q_2\{^k/_{x_2}\})]; \Psi; \emptyset)$ and $(\mathcal{E}_0; C[P_1[0] \mid P_2[0]]; \Phi; \emptyset)$ do not reveal key in $\{n, \mathsf{pk}(n), \mathsf{vk}(n) \mid n \in fn(P_1, P_2) \cap fn(Q_1, Q_2) \cap bn(C)\}$, and
– $(\mathcal{E}_0; C[\mathtt{new}\ k.(Q_1\{^k/_{x_1}\} \mid Q_2\{^k/_{x_2}\})]; \Psi; \emptyset)$ do not reveal $k, \mathsf{pk}(k), \mathsf{vk}(k)$, and
– $P_1/P_2/\Phi$ is a good key-exchange protocol under $\mathcal{E}_0$ and C, that implies in particular that $(\mathcal{E}_0; P_{\mathsf{good}}; \Phi)$ does not reveal bad where P_{good} is defined as follows:

$$\begin{array}{l}
P_{\mathsf{good}} = \mathtt{new}\, bad, d.\big(\\
\quad C[\mathtt{new}\, id.(P_1[\mathtt{out}(d, \langle x_1, id\rangle)] \mid P_2[\mathtt{out}(d, \langle x_2, id\rangle)])] \\
\quad \mid \mathtt{in}(d, x).\mathtt{in}(c, z). \\
\quad\quad \mathtt{if}\ z \in \{\mathsf{proj}_1(x), \mathsf{pk}(\mathsf{proj}_1(x)), \mathsf{vk}(\mathsf{proj}_1(x))\} \\
\quad\quad \mathtt{then}\, \mathtt{out}(c, bad)\big)
\end{array}$$

As done previously, relying on Lemma 26 (or Lemma 27 in case C is of the form $C'[!\_]$), we may assume that the trace under study is compatible. Applying Theorem 5, we deduce that there exists a biprocess SD' such that $SD \stackrel{\mathsf{tr}}{\Rightarrow}_{\mathsf{bi}} SD'$ with $\mathsf{fst}(SD') = S_n'^{\mathsf{v}}$, and static equivalence holds between the two frames issued from the biprocess SD'. Moreover, if we denote by Φ_S and Φ_D the respective frame of $\mathsf{fst}(SD')$ and $\mathsf{snd}(SD')$, we ensure that $\delta(\Phi_S\!\downarrow) = \Phi_D\!\downarrow$ (see Proposition 1).

Therefore, since D_n^{vseq} does not reveal the secret s, and we already proved that D_n^{vseq} does not reveal his assignment variables, then by Lemma 16, we can deduce that S_n^{v} does not reveal s, and so S does not reveal s either.