Oracle simulation: a technique for protocol composition with long term shared secrets

Hubert Comon, Charlie Jacomme, Guillaume Scerri

Oracle simulation: a technique for protocol composition with long term shared secrets

Hubert Comon[1], Charlie Jacomme[2], and Guillaume Scerri[3]

[1,2]LSV, CNRS & ENS Paris-Saclay & Inria & Université Paris-Saclay
[3]Université Versailles Saint-Quentin & Inria

August 10, 2020

Abstract

We provide a composition framework together with a variety of composition theorems allowing to split the security proof of an unbounded number of sessions of a compound protocol into simpler goals. While many proof techniques could be used to prove the subgoals, our model is particularly well suited to the *Computationally Complete Symbolic Attacker* (CCSA) model.

We address both sequential and parallel composition, with state passing and long term shared secrets between the protocols. We also provide with tools to reduce multi-session security to single session security, with respect to a stronger attacker. As a consequence, our framework allows, for the first time, to perform proofs in the CCSA model for an unbounded number of sessions.

To this end, we introduce the notion of $\mathcal{O}$-simulation: a simulation by a machine that has access to an oracle $\mathcal{O}$. Carefully managing the access to long term secrets, we can reduce the security of a composed protocol, for instance $P\|Q$, to the security of P (resp. Q), with respect to an attacker simulating Q (resp. P) using an oracle $\mathcal{O}$. As demonstrated by our case studies the oracle is most of the time quite generic and simple.

These results yield simple formal proofs of composed protocols, such as multiple sessions of key exchanges, together with multiple sessions of protocols using the exchanged keys, even when all the parts share long terms secrets (e.g. signing keys). We also provide with a concrete application to the SSH protocol with (a modified) forwarding agent, a complex case of long term shared secrets, which we formally prove secure.

Contents

Part I
The Framework

1 Introduction

This paper is concerned with the security proofs of composed protocols. This topic has been widely studied in the last two decades. For instance, Universal Composability (UC) and simulation based reductions [1–6] and other game-based composition methods [7–10] address this issue. While the former proceed in a more bottom-up manner (from secure components in

any environment, construct secure complex protocols), the latter proceed in a more top-down way: from the desired security of a complex protocol, derive sufficient security properties of its components. Such "top-down" proofs design allows more flexibility: the security requirements for a component can be weaker in a given environment than in an arbitrary environment. The counterpart is the lack of "universality": the security of a component is suitable for some environments only.

We follow the "top-down" approach. While we aim at designing a general methodology, our target is the management of formal security proofs in the Computationally Complete Symbolic Attacker (CCSA) model [11]. As a side result of our work, we provide with a way of proving the security of an arbitrary number of sessions (that may depend on the security parameter) in the CCSA model.

When trying to (de-)compose security properties, the main difficulty comes from the fact that different protocols may share some secrets. This is typically the case for multiple sessions of the same protocol, or for key exchange protocols, which result in establishing a shared secret that will be later used in another protocol. Protocols may also share long term secrets, for instance the same signing key may be used for various authentication purposes. Another example is the SSH protocol with the agent forwarding feature [12], which we will consider later. The forwarding feature allows to obtain, through previously established secure SSH connections, signatures of fresh material required to establish new connections. It raises a difficulty, as signatures with a long term secret key are sent over a channel established using the same long term secret key.

As far as we know, the existing composition results that follow the "top-down" approach cannot be used in situations where there is both a "state passing", as in key exchange protocols, and shared long term secrets. For instance, in the nice framework of [10], the same public key cannot be used by several protocols, a key point for reducing security of multiple sessions to security of one session.

When decomposing the security of a composed protocol into the security of its components, we would like to break a complex proof into simpler proofs, while staying in the same proof framework. This is also a difficulty since the attacker on a protocol component might use the other components: we need a proof with respect to a stronger attacker. In [10], such a strong attacker can be simulated by a standard one, because there is no shared long term secret.

1.1 Our contributions

We provide with a composition framework that reduces the security of a compound protocols to the security of its components. We allow both state passing and shared long term secrets. We stay in the same proof framework of the CCSA model.

The starting idea is simple: if we wish to prove the security of a composed protocol $P\|Q$, it is sufficient to prove the security of P against an attacker that may simulate Q, maybe with the help of an oracle. If $\overline{n}$ are the secrets shared by P and Q, this simulation has to be independent of the distribution of $\overline{n}$. This is actually an idea that is similar to the key-independence of [8].

Therefore, we first introduce the notion of $\mathcal{O}$-simulation, in which an oracle $\mathcal{O}$ holds the shared secrets: if Q is $\mathcal{O}$-simulatable and P is secure against an attacker that has access to $\mathcal{O}$, then $P\|Q$ is secure. Intuitively, $\mathcal{O}$ defines an interface through which the secrets can be used (e.g. obtaining signatures of only well tagged messages). $\mathcal{O}$ simulatable protocols conform to this interface.

We extend this basic block to arbitrary parallel and sequential compositions, as well as replication of an unbounded number of copies of the same protocol. In the latter case, the security of a single copy of P against an attacker that has access to an oracle allowing to simulate the other copies, requires to distinguish the various copies of a same protocol. In the universal composability framework, this kind of properties is ensured using explicit session identifiers. We rather follow a line, similar to [13], in which the session identifiers are implicit.

Our main composition Theorems are generic: the classical game based setting can be used to prove the subgoals. They are also specially well-suited for the CCSA model, which allows to complete computational proofs of real life protocols [14–16], while only relying on first order logic and cryptographic axioms. Many such axioms can easily be generalized so as to be sound with respect to an attacker that has access to oracles (we will see examples later).

A proof using such axioms is valid for an attacker who has access to an environment, while abstracting all the details of the environment and its interactions with the attacker. Moreover, as our reductions from one session to multiple sessions are uniform, we may now complete proofs in the CCSA model for a number of session that is parameterized by the security parameter. This was a limitation (and left as an open issue) in all previous CCSA papers.

We illustrate our composition results showing how to split the security of any (multi-session with shared long term secret) composed key exchange into smaller proofs. We then complete the formal proof of security of a Diffie-Hellman key exchange (ISO 9798-3 [17]) for any number of sessions in parallel.

We generalize the application to key exchanges performing key confirmations, i.e. using the derived key in the key exchange (as in TLS). The generalization is simple, which is a clue of the usability of our framework.

To illustrate the usability of our framework, we use all our results to prove the security of the SSH [12] protocol with a modified agent forwarding, a complex example of key exchange, with both key confirmation and long term shared secrets. The modification, which consists in the addition of a tag to specify if the signature was performed remotely, is necessary for the protocol to satisfy some natural security properties related to the agent forwarding.

1.2 Related Works

We introduce the composition problem through a process algebra: protocols are either building blocks (defined,e.g, with a transition system) or composed using parallel and sequential composition, and replication. This prevents from committing to any particular programming language, while keeping a clean operational semantics. This approach is also advocated in [10], which follows a similar approach. Other works on composition (e.g., [5, 7]) rely on specific execution models.

Our starting idea, to prove a component w.r.t. a stronger attacker that has access to the context, is not new. This is the basis of many works, including [8–10,18]. The main difference, that we wish to emphasize, is that these works do not support long term shared secrets, used in different components. Notably, the oracles of [10] are only used to decompose protocols with state passing. Our notion of simulatability allows sharing long term secret by granting the attacker access to oracles that depend on the secrets (for instance, signing oracles). It also allows a symmetric treatment for proofs of a protocol and proofs of its context.

For several specific problems, typically key exchanges, there are composition results allowing to prove independently the key exchange protocol and the protocol that uses the exchanged

key [8, 9, 13, 18, 19]. In such examples, the difficulty also comes from the shared secret, especially when there is a key confirmation step. In that case, the derived key is used for an integrity check, which is part of the key exchange. Then the property of the key exchange: "the key is indistinguishable from a random" does not hold after the key confirmation and thus cannot be used in the security proof of the protocol that uses this exchanged key. In [8], the authors define the notion of key independent reduction, where, if an attacker can break a protocol for some key distribution, he can break the primitive for the same distribution of the key. This is related to our notion of simulatability, as interactions with shared secrets are captured by an oracle for fixed values of the key, and thus attacks on the protocol for a fixed distribution are naturally translated into attacks against the primitive for the same distribution. Key exchanges with key confirmation are therefore a simple application of our composition results. Along the same line, [19] extends [18] to multi staged key exchanges, where multiple keys might be derived during the protocol. While we do not directly tackle this in our paper, our framework could be used for this case.

The authors of [9] also provide results allowing for the study of key renewal protocols (which we capture with the sequential replication Theorem), and has the advantage to be inside a mechanized framework, while we only cast our results inside a mechanizable framework. It does not however consider key confirmations.

The UC framework initiated by [1] and continued in [3, 4, 6] is a popular way of tackling composition. As explained above, this follows a "bottom-up" approach, in which protocols must be secure in any context, which often yield very strong security properties, some of which are not met in real life protocols. Moreover, to handle multiple sessions of a protocol using a shared secret, joint-state theorems are required. This requires a tagging mechanism with a distinct session identifier (sid) for each session. Relaxing this condition, the use of implicit session identifiers was established in [20] for the UC framework, ideas continued in [13] for Diffie-Hellman key exchanges, where they notably provide a proof of the ISO 9798-3 [17] protocol.

We do not consider a composition that is universal: it depends on the context. This allows us to relax the security properties regarding the protocol, and thus prove the compositional security of some protocols that cannot be proved secure in the UC sense. We also rely on implicit sids to prove the security of multiple sessions. Some limitations of the UC framework are discussed in [18, Appendix A].

In [5], the authors also address the flexibility of UC (or reactive simulatability) showing how to circumvent some of its limitations. The so-called "predicates" are used to restrict the order and contents of messages from environment and define a conditional composability. Assuming a joint-state conditional composability theorem, secret sharing between the environment and the protocol might be handled by restricting the accepted messages to the expected use of the shared secrets. However, the framework does not cover how to prove the required properties of (an instance of) the environment.

Protocol Composition Logic is a formal framework [21] designed for proving, in a "Dolev-Yao model", the security of protocols in a compositional way. Its computational semantics is very far from the usual game-based semantics, and thus the guarantees it provides [22] are unclear. Some limitations of PCL are detailed in [23].

The compositional security of SSH, in the sense of [18], has been studied in [24]. They do not consider however the agent forwarding feature. It introduces important difficulties since the key exchange is composed with a second key exchange that uses both the first derived key and the same long term secrets. SSH has also been studied, without agent forwarding, in [25],

where the implementation is derived from a secure modelling in CryptoVerif [26].

Summing up, our work is strongly linked to previous composition results and captures analogues of the following notions in our formalism: implicit disjointness of local session identifiers [20], single session games [18], key-independent reductions [8] and the classical proof technique based on pushing part of a protocol inside an attacker, as recently formalized in [10]. We build on all these works and additionally allow sharing long term secrets, thanks to a new notion of $\mathcal{O}$-simulatability. This fits with the CCSA model: the formal proofs of composed protocols are broken into formal proofs of components. All these features are illustrated by a proof of SSH with (a modified) agent forwarding.

2 Protocols and Indistinguishability

We first recall some features of the CCSA model. Although this model is not used until the case studies, it may be useful for an easier understanding of the protocol semantics.

2.1 Syntax and semantics of terms

To enable composition with long term shared secrets, we must be able to specify precisely the shared randomness between protocols. We use symbols from an alphabet of *names*, to represent the random samplings. The same symbol used twice represents the same (shared) randomness. Those names can be seen as pointers to a specific randomness, where all the randomness has been sampled upfront at the beginning of the protocol. This idea stems from the CCSA model [11], from which we re-use exactly the same term semantics. This is one of the reason why our results, while applicable in a broader context, fit naturally in the CCSA model. Let us recall the syntax and semantics of terms drawn from the CCSA model.

Syntax We use terms built over explicit names to denote messages computed by the protocol. The terms are defined with the following syntax:

$$\begin{array}{rcll} t & ::= & n & \text{names} \\ & | & n_{\vec{i}} & \text{indexed names} \\ & | & x & \text{variable} \\ & | & f(t_1, \ldots, t_n) & \text{operation of arity } n \end{array}$$

A key addition to the CCSA model is that some names can be indexed by sequences of index variables. This is necessary so that we may later on consider the replication of protocols. When a replicated protocol depends on a name n_i, the first copy (session) of the protocol uses n_1, the second n_2, ... Names without index models randomness shared by all sessions of the protocol. Variables are used to model the attacker inputs, and function symbols allows to model the cryptographic computations.

Semantics Terms are interpreted as bitstrings. As in the computational model, the interpretation depends on some security parameter η. As we assume that all the randomness is sampled at the beginning, the interpretation depends on an infinitely long random tape ρ_s. We then leverage the notion of a *cryptographic library*[1], that provides an interpretation for

[1] This corresponds in the CCSA model to the notion of functional model.

all names and function symbols. A cryptographic library $\mathcal{M}_f$ provides for each name n a Probabilistic Polynomial Time Turing Machine (PPTM for short) $\mathcal{A}_n$, that is given access to the random tape ρ_s. As an additional input, all machines will always be given the security parameter in unary. Each $\mathcal{A}_n$ extracts a bit-string of length η from the random tape. Different names extract non-overlapping parts of the random tape. In the interpretation, we give to all the PPTM the same random tape ρ_s, so each name is always interpreted with the same value in any term (and thus any protocol), and all names are interpreted independently.

$\mathcal{M}_f$ also provides for each function symbol f (encryption, signature,...) a PPTM $\mathcal{A}_f$, that must be deterministic. To model randomized cryptographic primitives, additional randomness must be given to the function symbol as extra names (cf. Example 2.1).

Given $\mathcal{M}_f$, the semantic mapping $[\![\cdot]\!]_{\rho_s}^{\eta,\sigma}$ evaluates its argument, a formal term, given an assignment σ of its variables to bit-strings and a random tape ρ_s. For instance, if n is a name, $[\![n]\!]_{\rho_s}^{\eta} = \mathcal{A}_n(1^\eta, \rho_s)$ (extracts a bit-string of length η from the random tape ρ_s) and $[\![\mathsf{sign}(x,k)]\!]_{\rho_s}^{\eta,\{x\mapsto m\}} = \mathcal{A}_{\mathsf{sign}}(m, \mathcal{A}_k(1^\eta, \rho_s))$. The details about the syntax and semantic of messages can be found in Appendix A.

2.2 Syntax of the protocols

The summary of the protocol syntax is given in Figure 1. An elementary protocol models a thread running on a specific computer. let denotes variable binding inside a thread, $\mathtt{in}(c,x)$ (resp. $(\mathtt{out}(c,m))$ denotes an input (resp. an output) of the thread over the channel c, where all channels are taken out of a set $\mathcal{C}$. For simplicity, channel identifiers are constants or indexed constants. In particular, they are known to the attacker. The if then else constructs denotes conditionals, $\mathbf{0}$ is a successfully terminated thread and $\perp$ is an aborted thread.

For protocols, our goal is to state and prove general composition results: we first consider sequential composition (the ; operator), where $\mathbf{0};P$ reduces to P, while $\perp;P$ reduces to $\perp$. In most cases, we will omit $\mathbf{0}$. We also consider parallel composition (the $\|$ operator), a fixed number N of copies running concurrently $\|^{i\leq N}$, as well as an arbitrary number of copies running concurrently $\|^i$. For instance, we can express a (two-parties) key-exchange consisting of an initiator I and a responder R with $I\|R$, the key exchange followed by a protocol using the exchanged key $(I;P^I)\|(R;P^R)$, as well as any number of copies of the resulting protocol running in parallel: $\|^i((I;P^I)\|(R;P^R))$. We can also consider an arbitrary iteration of a protocol, "$;^i$", which could be used for expressing, for instance, key renewal.

We provide in Appendix B a full definition of the protocol algebra. For generality, the full algebra is also parameterized by some atomic protocols, that can be used to easily extend the syntax.

We allow terms inside a protocol to depend on some free variables and, in this case, we denote $P(x_1,\ldots,x_n)$ a protocol, which depends on free variables $x_1,\ldots,x_n$. $P(t_1,\ldots,t_n)$ denotes the protocol obtained when instantiating each x_i by the term t_i.

We denote $\mathcal{N}(P)$ (resp $\mathcal{C}(P)$) the set of names (resp. channel names) of P.

Example 2.1. Given a randomized encryption function $\mathtt{enc}$, we let $P(c,x_1,x_2)$ be the protocol $\mathtt{in}(c,x).\mathtt{out}(c,\mathtt{enc}(x,x_1,x_2))$. Given names sk,r representing respectively a secret key and a random seed, $E_N := \|^{i\leq N} P(c_i,r_i,sk)$ is then the protocol allowing the attacker to obtain cyphertexts for an unknown secret key sk. Unfolding the definitions, we get:

$$E_N := P(c_1,r_1,sk)\|\ldots\|P(c_n,r_n,sk)$$

The generalization giving access to encryption for five secret keys is expressed with $\|^i\|^{j\leq 5} P(c_{j,i},r_{j,i},sk_j)$.

elementary protocols:			
P_{el}	::=	$\mathsf{let}\ x = t\ \mathsf{in}\ P_{el}$	variable binding
	|	$\mathtt{in}(c, x).P_{el}$	input
	|	$\mathtt{out}(c, m).P_{el}$	output
	|	$\mathsf{if}\ s = t\ \ \mathsf{then}\ P_{el}\ \mathsf{else}\ P_{el}$	conditional
	|	$\mathbf{0}$	success
	|	$\bot$	failure
protocols:			
P, P'	::=	P_{el}	
	|	$P_{el}; P$	sequential composition
	|	$P \| P'$	parallel composition
	|	$\|^{i \leq N} P$	bounded replication
	|	$\|^{i} P$	unbounded replication

Figure 1: The protocol algebra

$$\frac{\phi, (P, \sigma) \underset{\mathcal{A}}{\rightarrow} \phi', (P', \sigma')}{\phi, (P; Q, \sigma) \underset{\mathcal{A}}{\rightarrow} \phi', (P'; Q, \sigma')} \qquad \frac{}{\phi, (\mathbf{0}; Q, \sigma) \underset{\mathcal{A}}{\rightarrow} \phi, (Q, \sigma)}$$

$$\frac{\phi, (P, \sigma) \underset{\mathcal{A}}{\rightarrow} \phi', (P', \sigma')}{\phi, (P, \sigma) \| E \underset{\mathcal{A}}{\rightarrow} \phi', (P', \sigma') \| E}$$

Figure 2: Operational Semantics (excerpt)

2.3 Semantics of the protocols

We give here some essential features of the formal execution model, which we need to formalize our composition results.

A (global) state of a protocol consists in a *frame*, which is a sequence of bit-strings modelling the current attacker knowledge, and a finite multiset of pairs (P, σ), where P is a protocol and σ is a local binding of variables. Intuitively, each of the components of the multiset is the current state of a running thread. We write such global states $\phi, (P_1, \sigma_1) \| \cdots \| (P_n, \sigma_n)$.

The transition relation between global states is parameterized by an attacker $\mathcal{A}$ who interacts with the protocol, modelled as a PPTM with its dedicated random tape ρ_r. The attacker chooses which of the threads is going to move and computes, given ϕ, the input to that thread. In the following, the configuration of the protocol and the security parameter are (also) always given to the attacker, which we do not make explicit for simplicity.

We give some of the rules describing the Structural Operational Semantics in Figure 2. The full semantics can be found in Appendix B. The transition relation $\underset{\mathcal{A}}{\rightarrow}$ between configurations depends on the attacker $\mathcal{A}$, the security parameter η and the random samplings ρ_s (to interpret terms) and ρ_r (the randomness of the attacker). In $P; Q$, P has to be executed first. When it is completed (state $\mathbf{0}$), then the process can move to Q, inheriting the variable bindings from P. If P is not waiting for an input from the environment, it can move independently from any of the other parallel processes.

The semantics of inputs (not detailed for simplicity) reflects the interactions with the attacker. $\mathcal{A}$ computes the input to the protocol, given a frame ϕ and its own random tape ρ_r. Therefore transitions depend not only on the attacker machines, but also[2] on the name samplings ρ_s (secret coins) and ρ_r (attacker's coins).

Example 2.2. Continuing Example 2.1, the initial configuration corresponding to E_2 is $\emptyset, (P(c_1, r_1, sk), \emptyset) \| (P(c_2, r_2, sk), \emptyset)$, where the attacker knowledge is empty and no local variables are bound. We consider one of the possible reductions, for some attacker $\mathcal{A}$ that first sends a message over channel c_1 and then c_2:

$$
\begin{aligned}
&\emptyset, (P(c_1, r_1, sk), \emptyset) \| (P(c_2, r_2, sk), \emptyset) \\
&\quad \underset{\mathcal{A}}{\longrightarrow} \emptyset, (\mathtt{out}(c_1, \mathtt{enc}(x, r_1, sk), \{x \mapsto m\}) \| (P(c_2 r_2, sk), \emptyset) \\
&\qquad\qquad m = \mathcal{A}(\emptyset, \rho_r) \text{ is the first input message computed by the attacker} \\
&\quad \underset{\mathcal{A}}{\longrightarrow} \phi, (P(c_2, r_2, sk), \emptyset) \\
&\qquad\qquad \phi = [\![\mathtt{enc}(x, r_1, sk)]\!]_{\rho_s}^{\eta, \{x \mapsto m\}} \text{ is the interpretation of the output received by the attacker} \\
&\quad \underset{\mathcal{A}}{\longrightarrow} \phi, (\mathtt{out}(c_2, \mathtt{enc}(x, r_1, sk), \{x \mapsto m_2\}) \\
&\qquad\qquad m_2 = \mathcal{A}(\phi, \rho_r) \text{ is the second input message computed by the attacker} \\
&\quad \underset{\mathcal{A}}{\longrightarrow} (\phi, [\![\mathtt{enc}(x, r_2, sk)]\!]_{\rho_s}^{\eta, \{x \mapsto m_2\}}), \mathbf{0}
\end{aligned}
$$

We assume *action determinism* of the protocols [27]: given an input message on a given channel, if the current state is

$$\phi, (P_1, \sigma_1) \| \cdots \| (P_n, \sigma_n),$$

at most one of the P_i may move to a non-abort state. This means that each thread checks first that it is the intended recipient of the message. This also means that each output has to be triggered by an input signal: none of the P_i starts with an output action. We remark that in practice, protocols are action determinate.

For replicated protocols $\|^{i \leq N} P$ or $\|^i P$, the names in P that are indexed by the variable i are renamed as follows: $\|^{i \leq N} P$ is the protocol $P\{i \mapsto 1\} \| \ldots \| P\{i \mapsto N\}$ and

$$\phi, (\|^i P, \sigma) \| E \underset{\mathcal{A}}{\longrightarrow} \phi, (\|^{i \leq \mathcal{A}(\rho_r, \phi)} P, \sigma) \| E.$$

In other words, the attacker chooses how many copies of P will be considered, which may depend, in particular, on the security parameter. $\mathcal{A}(\rho_r, \phi)$ must be a natural number in unary.

2.4 Stateless Oracle Machines

For reasons that have been explained in the introduction, we wish to extend the semantics of protocols and their indistinguishability to attackers that have access to an additional stateless oracle. At this stage, we need stateless oracles in order to be compositional. Let us explain

[2] They actually also depend on the oracle's coins, when $\mathcal{A}$ is interacting with an external oracle, which we explain later.

this. Assume we wish to prove a property of R in the context $P\|Q\|R$. The idea would be to prove R, interacting with an attacker that simulates $P\|Q$. This attacker is itself a composition of an attacker that simulates P and an attacker that simulates Q. The protocols P, Q, R share primitives and secrets, hence the simulation of P, Q requires access to an oracle that holds the secrets. If such an oracle were to be stateful, we could not always build a simulator for $P\|Q$ from simulators of P, Q respectively, since oracle replies while simulating Q could depend on oracle queries made while simulating P, for instance.

The oracles depend on a security parameter η (that will not always be explicit), (secret) random values and also draw additional coins: as a typical example, a (symmetric key) encryption oracle will depend on the key k and use a random number r to compute $\mathtt{enc}(m, r, k)$ from its query m. Therefore, the oracles can be seen as deterministic functions that take two random tapes as inputs: ρ_s for the secret values and $\rho_{\mathcal{O}}$ for the oracle coins.

Formally, oracles take as input tuples $(\overline{m}, r, s)$ where $\overline{m}$ is a finite sequence of bitstrings, r is a handle for a random value and s is a handle for a secret value. r and s are respectively used to extract the appropriate parts of $\rho_{\mathcal{O}}, \rho_s$ respectively, in a deterministic way: the randomness extracted from $\rho_{\mathcal{O}}$ is uniquely determined by $\overline{m}, r, s$ and the extractions for different values do not overlap.

In what follows, we only consider oracles that are consistent with a given cryptographic library $\mathcal{M}_f$. Such oracles only access ρ_s through some specific names. This set of names is called the *support* of the oracle.

Example 2.3. An encryption oracle for the key k (corresponding to the handle "1"), successively queried with $(m, 1, 1), (m', 2, 1), (m, 3, 1), (m, 1, 1), (m', 2, 2), \ldots$ will produce respectively the outputs $\mathtt{enc}(m, r_1, k)$, $\mathtt{enc}(m', r_2, k)$, $\mathtt{enc}(m, r_3, k)$, $\mathtt{enc}(m, r_1, k)$, $\perp, \ldots$ Here r_1, r_2, r_3 are non-overlapping parts of $\rho_{\mathcal{O}}$ (each of length η). The support of this oracle is $\{k\}$.

The formal definition of stateless oracles is a bit involved, notably to formally specify the randomness extraction. This construction is required to ensure the determinism of the oracles. Determinism is required to build a single simulator for two parallel protocols from the individual simulators for the two protocols.

For instance, for an oracle performing randomized encryption, rather than always encrypting with a fresh nonce, this system allows multiple attackers to obtain an encryption of a message with the same random.

Definition 1 ((Stateless) Oracle)**.** An *oracle* $\mathcal{O}$ is a triple of functions that have the following inputs

- a sequence of bitstrings $\overline{w} \in (\{0,1\}^*)^n$ and two bitstrings r, s: the query, consisting of an *input query* $\overline{w}$, an *input tag* r, an *input key* s;
- a random tape ρ_s for the (secret) random values;
- the security parameter η;
- a random tape $\rho_{\mathcal{O}}$ for the oracle's coins.

The first function assigns to each $\overline{w}, s, r$ an integer $n(\overline{w}, s, r) \in \mathbb{N}$ and is assumed injective. $n(\overline{w}, s, r)$ is used to extract a substring $e_1(n(\overline{w}, s, r), \eta, \rho_{\mathcal{O}})$ from $\rho_{\mathcal{O}}$, which is uniquely determined by the input. We assume that the length of the substring extracted by e_1 only depends on η, and substrings extracted with e_1 are disjoint for different values of n.

The second function e_2 assigns to each s a sequence $\overline{p}(s)$ of natural numbers, that are used to extract secret values from ρ_s: $e_2(s, \eta, \rho_s)$ is a sequence of bitstrings. It is also assumed to be injective.

The third function takes $\eta, \overline{w}$, r, s, $e_1(n(\overline{w}, s, r), \eta, \rho_{\mathcal{O}})$, $e_2(s, \eta, \rho_s)$ as input and returns a result (a bitstring) or a failure message.

Example 2.4. Expanding upon Example 2.3, the encryption oracle is given by the triple of functions (e_1, e_2, e_3) such that:

- $e_1(n(\overline{w}, s, r), \eta, \rho_{\mathcal{O}})$ extracts the substring r at position range $[n(\overline{w}, s, r) \times \eta, (n(\overline{w}, s, r) + 1) \times \eta]$ from $\rho_{\mathcal{O}}$.
- $e_2(s, \eta, \rho_s) = \begin{cases} [\![k]\!]^{\eta}_{\rho_s} \text{ if } s = 1 \\ 0 \text{ else} \end{cases}$
- $e_3(\eta, \overline{w}, r, s, e_1(n(\overline{w}, s, r), \eta, \rho_{\mathcal{O}}), e_2(s, \eta, \rho_s)) = [\![\mathtt{enc}(y, r, x)]\!]^{\eta}_{\{y \mapsto \overline{w}, r \mapsto r, x \mapsto e_2(s, \eta, \rho_s)}$

Given η, and a sequence of bitstrings m, we call r_1 the sequence of bitstrings at position range $[n(m, 1, 1) \times \eta, (n(m, 1, 1) + 1) \times \eta]$ from $\rho_{\mathcal{O}}$. Then, on input $(m, 1, 1)$, $e_1(n(m, 1, 1), \eta, \rho_{\mathcal{O}}) = r_1$, $e_2(1, \eta, \rho_s) = [\![k]\!]^{\eta}_{\rho_s}$ and the oracle returns $e_3(\eta, m, 1, 1, r_1, [\![k]\!]^{\eta}_{\rho_s}) = [\![\mathtt{enc}(y, r, k)]\!]^{\eta}_{y \mapsto m, r \mapsto r_1}$.

An oracle machine (PPTOM) is a PPTM, equipped with an additional tape, on which the queries to the oracle are written and from which the oracle replies are read. We often write explicitly the machine inputs, as in $\mathcal{A}^{\mathcal{O}(\rho_s, \rho_{\mathcal{O}})}(\omega, \rho_r)$, where ω is the input data of $\mathcal{A}$, ρ_r is its random tape and $\rho_s, \rho_{\mathcal{O}}$ are the random tapes accessible to the oracle. These definitions extend to multiple oracles $\langle \mathcal{O}_1, \ldots, \mathcal{O}_n \rangle$, prefixing the query with an index in $\{1, \ldots, n\}$.

Definition 2. A *Probabilistic Polynomial Time Oracle Machine* (PPTOM) is a Turing machine denoted by $\mathcal{A}^{\mathcal{O}}$ and equipped with:

- an input/working/output tape (as usual; it is read/write);
- a read-only random tape ρ_r (attacker's coins);
- an oracle input tape $\rho_{\mathcal{O}}$;
- an oracle output tape, which is read-only.
- an oracle read-only random tape ρ_s (not accessible by the Turing Machine);

Note that once the oracle's random tape is fixed, we ensure that all our oracles are deterministic.

2.5 Computational indistinguishability

To define the classical notion of indistinguishability, we describe how protocols may be seen as oracles, that an attacker can interact with. Given a protocol P and a cryptographic library $\mathcal{M}_f$, the oracle $\mathcal{O}_P$ is an extension of the previous oracles: it takes as an additional input an history tape that records the previous queries. Given a query m with history h (now the components r, s are useless), the oracle replies what would be the output of P, given the

successive inputs h, m. It also appends the query m to the history tape. The formal definition of protocol oracles can be found in Appendix B.3.

The machines that interact with $\mathcal{O}_P$ are also equipped with the history tape that is read-only: the history can only be modified by the oracle. Since P may use secret data, the oracle may access a secret tape ρ_s; this will be explicit.

An oracle may implement multiple parallel protocols: the oracle $\mathcal{O}_{\langle P_1,...,P_n\rangle}$ first checks which P_i is queried (there is at most one such i, by action determinism) and then replies as $\mathcal{O}_{P_i}$.

Finally, we may consider oracles that combine protocols oracles and stateless oracles. $\mathcal{A}^{\langle\mathcal{O}_1,...,\mathcal{O}_m\rangle,\langle\mathcal{O}_{P_1},...,\mathcal{O}_{P_n}\rangle}$ is also written $\mathcal{A}^{\mathcal{O}_1,...,\mathcal{O}_m,\mathcal{O}_{P_1},...,\mathcal{O}_{P_n}}$.

Definition 3. Given a cryptographic library $\mathcal{M}^f$, an oracle $\mathcal{O}$ and protocols $P_1, \ldots, P_n, Q_1, \ldots, Q_n$, we write $\mathcal{A}^{\mathcal{O},\mathcal{O}_{P_1?Q_1},...,\mathcal{O}_{P_n?Q_n}} \prec \epsilon$ if for every polynomial time oracle Turing machine $\mathcal{A}^{\mathcal{O}}$,

$$\begin{aligned}|&\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}}),\mathcal{O}_{P_1}(\rho_s),...,\mathcal{O}_{P_n}(\rho_s)}(\rho_r,1^\eta)=1\}\\ &-\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}}),\mathcal{O}_{Q_1}(\rho_s),...,\mathcal{O}_{Q_n}(\rho_s)}(\rho_r,1^\eta)=1\}|\end{aligned}$$

is negligible in η. We will write $P \cong_{\mathcal{O}} Q$ for $\mathcal{A}^{\mathcal{O},\mathcal{O}_{P?Q}} \prec \epsilon$.

Example 2.5. For $i \in \{1, 2\}$, the protocol P_i is defined with the single transition:

$$q, \{x_1, ..., x_n\} \xrightarrow{x_{n+1}=(m_1,m_2)} (q, enc(m_i, sk), \{x_1, ..., x_n, x_{n+1}\})$$

The protocol expects to receive a couple as input, and will output either the left message or the right message using some secret key. $P_1 \cong_{\mathcal{O}} P_2$ then captures the fact that an attacker with oracle $\mathcal{O}$ has a negligible probability to win the IND-CPA game.

By construction, indistinguishability is compatible with our constructions for protocols in parallel and multiple protocol oracles. Indeed the oracle protocol for $P\|Q$ behaves exactly the same as the two oracle $\mathcal{O}_P$ and $\mathcal{O}_Q$ in parallel.

Lemma 4. *For protocols P, Q, A, B, an oracle $\mathcal{O}$, and a list $\mathcal{O}_l$ of protocol oracles,*

$$\mathcal{A}^{\mathcal{O},\mathcal{O}_{(A\|P)?(B\|Q)}} \prec \epsilon \Leftrightarrow \mathcal{A}^{\mathcal{O},\mathcal{O}_l,\mathcal{O}_{A?B},\mathcal{O}_{P?Q}} \prec \epsilon$$

3 Simulatability

We define a notion of "perfect" simulation, where a protocol depends on some secrets that the attacker can only access through an oracle, and an attacker must be able to produce exactly the same message as the protocol. This means that an attacker, given access $\mathcal{O}$ but not to a set of secrets $\overline{n}$, can completely simulate the protocol P (using $\mathcal{O}$ to have a partial access to the secrets), i.e., produce exactly the same distribution of message.

Formally, given a set of names $\overline{n}$, an oracle $\mathcal{O}$ and a protocol P. We say that $\nu\overline{n}.P$ is $\mathcal{O}$-simulatable, if there exists a PTOM $\mathcal{A}^{\mathcal{O}}$ such that for any attacker $\mathcal{B}^{\mathcal{O}}$, the sequences of messages produced by $\mathcal{B}^{\mathcal{O},\mathcal{O}_P}$ has exactly the same probability distribution as the on produced by $\mathcal{B}^{\mathcal{O}}$ interacting with $\mathcal{A}^{\mathcal{O}}$ instead of $\mathcal{O}_P$.

Assume that $Q \cong_{\mathcal{O}} R$ and $\nu\overline{n}.P$ is $\mathcal{O}$-simulatable, where $\overline{n}$ contains the secrets shared by P, Q and R. Any distinguisher against $Q \cong_{\mathcal{O}} R$ can also produce any message that would produce P in this context, and can therefore be transformed into a distinguisher against $Q\|P \cong_{\mathcal{O}} R\|P$. In other terms, $Q \cong_{\mathcal{O}} R$ and $\nu\overline{n}.P$ is $\mathcal{O}$-simulatable implies that $Q\|P \cong R\|P$.

3.1 Protocol Simulation

The goal in the rest of the paper is to use this notion of simulatability to obtain composability results. Suppose one wants to prove $P\|Q \cong P\|R$, knowing that $Q \cong_{\mathcal{O}} R$ and P is $\mathcal{O}$-simulatable. The way to obtain a distinguisher for $Q \cong_{\mathcal{O}} R$ from one on $P\|Q \cong P\|R$ is to "push" the (simulated version) of P within the distinguisher. A protocol P is then simulatable if there exists a simulator $\mathcal{A}^{\mathcal{O}}$ that can be "pushed " in any distinguisher $\mathcal{D}$. We formalize this construction below, where a protocol is simulatable if and only if any distinguisher $\mathcal{D}$ behaves in the same way if the protocol oracle $\mathcal{O}_P$ is replaced by its simulator $\mathcal{A}^{\mathcal{O}}$. We define formally $\mathcal{D}[\mathcal{A}^{\mathcal{O}}]^{\mathcal{O}}$ the replacement of $\mathcal{O}_P$ in $\mathcal{D}^{\mathcal{O},\mathcal{O}_P}$.

Definition 5. Given an oracle $\mathcal{O}$, a cryptographic library $\mathcal{M}_f$, a protocol P, PTOMs $\mathcal{D}^{\mathcal{O},\mathcal{O}_P}(\rho_{r_{\mathcal{D}}}, 1^\eta)$ and $\mathcal{A}^{\mathcal{O}}(\cdots, 1^\eta)$, we define $\mathcal{D}[\mathcal{A}^{\mathcal{O}}]^{\mathcal{O}}(\rho_r, 1^\eta)$ as the PTOM that:

1. Splits its random tape ρ_r into ρ_{r_1}, ρ_{r_2}
2. Simulates $\mathcal{D}^{\mathcal{O},\mathcal{O}_P}(\rho_{r_2}, 1^\eta)$ by replacing every call to $\mathcal{O}_P$ with a computation of $\mathcal{A}^{\mathcal{O}}$: each time $\mathcal{D}$ enters a state corresponding to a call to $\mathcal{O}_P$, $\mathcal{D}[\mathcal{A}^{\mathcal{O}}]$ appends the query m to a history θ (initially empty), executes the subroutine $\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\rho_{r_1}, \theta, 1^\eta)$ and behaves as if the result of the subroutine was the oracle reply.
3. Prefixes each random handle of an oracle call of $\mathcal{D}$ with 0 and random handle of an oracle call of $\mathcal{A}$ with 1.
4. Outputs the final result of $\mathcal{D}$.

$\mathcal{D}[\mathcal{A}^{\mathcal{O}}]^{\mathcal{O}}$ must simulate $\mathcal{A}^{\mathcal{O}}$ and $\mathcal{D}$ so that they do not share randomness. To this end, $\mathcal{D}[\mathcal{A}^{\mathcal{O}}]^{\mathcal{O}}$ first splits its random tape ρ_r into ρ_{r_1} (playing the role of $\rho_{\mathcal{O}}$) and ρ_{r_2} (playing the role of $\rho_{\mathcal{D}}$). The oracle queries are prefixed by distinct handles for the same reason. $\mathcal{D}^{\mathcal{O},\mathcal{O}_P}$ has access to the shared secrets via both $\mathcal{O}$ and $\mathcal{O}_P$, while $\mathcal{D}[\mathcal{A}^{\mathcal{O}}]^{\mathcal{O}}$ only has access to them through the oracle $\mathcal{O}$. Remark that if $\mathcal{A}^{\mathcal{O}}$ and $\mathcal{D}^{\mathcal{O},\mathcal{O}_P}$ has a run-time polynomially bounded, so does $\mathcal{D}[\mathcal{A}^{\mathcal{O}}]^{\mathcal{O}}$.

To define the central notion of $\mathcal{O}$-simulatability, the distribution produced by any distinguisher interacting with the simulator must be the same as the distribution produced when it is interacting with the protocol. However, as we are considering a set of shared secrets $\overline{n}$ that might be used by other protocols, we need to ensure this equality of distributions for any fixed concrete value $\overline{v}$ of the shared secrets. Then, even if given access to other protocols using the shared secrets, no adversary may distinguish the protocol from its simulated version.

Definition 6. Given an oracle $\mathcal{O}$ with support $\overline{n}$, a cryptographic library $\mathcal{M}^f$, a protocol P, a sequence of names $\overline{n}$, then, $\nu\overline{n}.P$ is $\mathcal{O}$-*simulatable* if and only if there exists a PTOM $\mathcal{A}_P^{\mathcal{O}}$ such that for every PTOM $\mathcal{D}^{\mathcal{O},\mathcal{O}_P}$, for every η, every $\overline{v} \in (\{0,1\}^\eta)^{|\overline{n}|}, c \in \{0,1\}^\star$,

$$\begin{aligned}&\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P}(\rho_r, 1^\eta) = c \mid [\![\overline{n}]\!]^\eta_{\rho_s} = \overline{v}\}\\ &\quad = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}_P^{\mathcal{O}}]^{\mathcal{O}}(\rho_r, 1^\eta) = c \mid [\![\overline{n}]\!]^\eta_{\rho_s} = \overline{v}\}\end{aligned}$$

Note that our definition of simulatability is a very strong one as it requires a perfect equality of distributions, as opposed to computational indistinguishability. This is intuitively what we want: $\mathcal{O}$-simulation expresses that P only uses the secrets in $\overline{n}$ as $\mathcal{O}$ does. This notion is not intended to capture any security property.

In practice, let us consider the security property $P\|Q \cong P\|Q'$, where P is simulatable by $\mathcal{A}_P^{\mathcal{O}}$. The idea of the later composition result is that an attacker $\mathcal{D}$ that distinguishes between $\mathcal{D}^{\mathcal{O},\mathcal{O}_P,\mathcal{O}_Q}$ and $\mathcal{D}^{\mathcal{O},\mathcal{O}_P,\mathcal{O}_{Q'}}$ can be turned into an attacker that distinguishes between $\mathcal{D}[\mathcal{A}_P^{\mathcal{O}}]^{\mathcal{O},\mathcal{O}_Q}$ and $\mathcal{D}[\mathcal{A}_P^{\mathcal{O}}]^{\mathcal{O},\mathcal{O}_{Q'}}$. Notice that here, Q and P may share some secrets, and their distributions are not independent. The intuition is that Q is fixing a specific value for the shared name between P and Q, and P then needs to be simulatable for this fixed value. This is why the notion of simulatability asks that a protocol is simulatable for any fixed value of a set of secret names. The formalization of this proof technique is given by the following Proposition.

Proposition 7. *Given an oracle $\mathcal{O}$ with support $\overline{n}$, a cryptographic library $\mathcal{M}_f$, protocols P, Q such that $\mathcal{N}(P) \cap \mathcal{N}(Q) \subseteq \overline{n}$, then, for any PTOM $\mathcal{A}_P^{\mathcal{O}}$, $\nu\overline{n}.P$ is $\mathcal{O}$-simulatable with $\mathcal{A}_P^{\mathcal{O}}$ if and only if for every PTOM $\mathcal{D}^{\mathcal{O},\mathcal{O}_P,\mathcal{O}_Q}$, for every η, every $\overline{v} \in (\{0,1\}^\eta)^{|\overline{n}|}, c \in \{0,1\}^\star$,*

$$\begin{aligned}&\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P,\mathcal{O}_Q}(\rho_r, 1^\eta) = c \mid [\![\overline{n}]\!]^\eta_{\rho_s} = \overline{v}\}\\ &\quad = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}_P^{\mathcal{O}}]^{\mathcal{O},\mathcal{O}_Q}(\rho_r, 1^\eta) = c \mid [\![\overline{n}]\!]^\eta_{\rho_s} = \overline{v}\}\end{aligned}$$

It then implies that:

$$\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P,\mathcal{O}_Q}(\rho_r, 1^\eta) = c\} = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}_P^{\mathcal{O}}]^{\mathcal{O},\mathcal{O}_Q}(\rho_r, 1^\eta) = c\}$$

While this Definition intuitively captures the proof technique used to allow composition, it does not provide insight about how to prove the simulatability. Another equivalent definition states that a protocol is simulatable if there exists a simulator that can produce exactly the same distribution of messages as the protocol interacting with any attacker. We formalize in the following this second Definition, and prove that the two Definitions are equivalent, which also yields the proof of Proposition 7.

For this second Definition of simulation to be realizable, we need to ensure that simulator's oracle calls and attacker's oracle calls use a disjoint set of random coins for the oracle randomness. We thus assume, w.l.o.g., that the random handles r of simulator's queries are prefixed by 1. This ensures that, as long as adversaries only make oracle calls prefixed by 0 (this can be assumed w.l.o.g. since it only constrains the part of the oracle's random tape where the randomness is drawn) the oracle randomness used by the simulator is not used by the adversary. We provide later in Example 3.2 a complete example illustrating both simulation and the need of the prefix and a formal definition of prefixed models.

Definition 8. Given a cryptographic library $\mathcal{M}_f$, a sequence of names $\overline{n}$, an oracle $\mathcal{O}$ and a protocol P, we say that $\nu\overline{n}.P$ is *$\mathcal{O}$-simulatable* if the support of $\mathcal{O}$ is $\overline{n}$ and there is a PTOM $\mathcal{A}^{\mathcal{O}}$ (using random handles prefixed by 0) such that, for every $c \in \{0,1\}^\star$, for every $\overline{v} \in (\{0,1\}^\eta)^{|\overline{n}|}$, for every $m \geq 1$, for every PTOM $\mathcal{B}^{\mathcal{O}}$ (using random handles prefixed by 1),

$$\begin{aligned}&\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\rho_{r_1}, \theta_m^1, 1^\eta) = c \mid [\![\overline{n}]\!]^\eta_{\rho_s} = \overline{v}\}\\ &\quad = \quad \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{O}_P(\rho_s, \theta_m^2) = c \mid [\![\overline{n}]\!]^\eta_{\rho_s} = \overline{v}\}\end{aligned}$$

where

$$\begin{aligned}\phi_{k+1}^2 &= \phi_k^2, \mathcal{O}_P(\rho_s, \theta_k^2)\\ \phi_{k+1}^1 &= \phi_k^1, \mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\rho_{r_1}, \theta_k^1, \eta)\\ \theta_{k+1}^i &= \theta_k^i, \mathcal{B}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\rho_{r_2}, \eta, \phi_{k+1}^i)\end{aligned}$$

for $0 \leq k < m$ and $\phi_0 = \emptyset$, $\theta_0 = \mathcal{B}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\rho_{r_2}, \eta, \emptyset)$.

The machine $\mathcal{A}^{\mathcal{O}}$ can be seen as the simulator, while $\mathcal{B}$ is an adversary that computes the inputs: the definition states that there is a simulator, independently of the adversary. We asks for equality of distributions, between the sequence of messages θ^2, corresponding to the interactions of $\mathcal{B}^{\mathcal{O}}$ with $\mathcal{O}_P$, and the sequence of messages θ^1, corresponding to the interactions of $\mathcal{B}^{\mathcal{O}}$ with $\mathcal{A}^{\mathcal{O}}$.

Note that our definition of simulatability is a very strong one as it requires a perfect equality of distributions, as opposed to computational indistinguishability. This is intuitively what we want: $\mathcal{O}$-simulation expresses that P only uses the shared secrets as $\mathcal{O}$ does. This notion is not intended to capture any security property.

The two definitions are indeed equivalent. To prove this, a first technical Lemma is required. It shows that $\mathcal{O}$-simulation, whose definition implies the identical distributions of two messages produced either by the simulator or by the oracle, implies the equality of distributions of message sequences produced by either the oracle or the simulator. It is proved essentially via an induction on the length of the sequence of messages. For any sequence of names $\overline{n}$ and parameter η, we denote $D^{\eta}_{\overline{n}} = \{[\![\overline{n}]\!]^{\eta}_{\rho_s} | \rho_s \in \{0,1\}^{\omega}\}$ the set of possible interpretations of $\overline{n}$. We reuse the notations of Definition 8.

Lemma 9. *Given a cryptographic library $\mathcal{M}^f$, a sequence of names $\overline{n}$, an oracle $\mathcal{O}$ with support $\overline{n}$ and a protocol P, that is* $\mathcal{O}$-simulatable *with $\mathcal{A}^{\mathcal{O}}$, we have, for every $\overline{x}, \overline{y}, c, r_2, r_{\mathcal{B}} \in \{0,1\}^{\star}$, every $\overline{v} \in D^{\eta}_{\overline{n}}$, for every $m \geq 1$, for every PTOM $\mathcal{B}^{\mathcal{O}}$ (using tags prefixed by 1):*

$$\begin{aligned}&\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta^1_m = \overline{x}, \phi^1_m = \overline{y} |\ [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}, \rho^{\mathcal{B}}_{\mathcal{O}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\}\\ &\quad = \quad \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta^2_m = \overline{x}, \phi^2_m = \overline{y} |\ [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}, \rho^{\mathcal{B}}_{\mathcal{O}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\}\end{aligned}$$

where we split $\rho_{\mathcal{O}}$ into $\rho^{\mathcal{A}}_{\mathcal{O}} \uplus \rho^{\mathcal{B}}_{\mathcal{O}}$ such that $\mathcal{O}$ called by $\mathcal{B}$ only accesses $\rho^{\mathcal{B}}_{\mathcal{O}}$ and $\mathcal{O}$ called by $\mathcal{A}$ only accesses $\rho^{\mathcal{A}}_{\mathcal{O}}$ (which is possible thanks to the distinct prefixes).

We now prove that Definition 8 implies Definition 5, i.e that the simulatability implies that we can replace a protocol oracle by its simulator.

Lemma 10. *Given an oracle $\mathcal{O}$ (with support $\overline{n}$), a cryptographic library $\mathcal{M}_f$, a sequence of names $\overline{n}$, P, Q protocols, such that $\nu\overline{n}.P$ is $\mathcal{O}$-simulatable in the sense of Definition 8 with $\mathcal{A}^{\mathcal{O}}_P$ and $\mathcal{N}(P) \cap \mathcal{N}(Q) \subseteq \overline{n}$ then, for every PTOM $D^{\mathcal{O},\mathcal{O}_P,\mathcal{O}_Q}$ (prefixed by 1), every η, every $\overline{v} \in D^{\eta}_{\overline{n}}$ and every $c \in \{0,1\}^{\star}$,*

$$\begin{aligned}&\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P,\mathcal{O}_Q}(\rho_r, 1^{\eta}) = c \mid [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}\}\\ &\quad = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}^{\mathcal{O}}_P]^{\mathcal{O},\mathcal{O}_Q}(\rho_r, 1^{\eta}) = c \mid [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}\}\end{aligned}$$

The idea is to use the definition of $\mathcal{O}$-simulatability, using a PTOM $\mathcal{B}^{\mathcal{O}}$ that behaves exactly as $\mathcal{D}$ when it computes the next oracle queries from the previous answers. The difficulty is that $\mathcal{D}$ may call the oracle $\mathcal{O}_Q$, while $\mathcal{B}$ has no access to this oracle. We know however that shared names are included in $\overline{n}$, whose sampling can be fixed at once (thanks to the definition of $\mathcal{O}$-simulation). The other randomness in Q can be drawn by $\mathcal{B}$ from ρ_r, without changing the distribution of $\mathcal{O}_Q$'s replies.

Proof. Fix η and the interpretation $[\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}$.

Given $\mathcal{D}$, we let $\mathcal{D}_m$ be the machine that behaves as $\mathcal{D}$, however halting after m calls to $\mathcal{O}_P$ (or when $\mathcal{D}$ halts if this occurs before the mth call) and returning the last query to $\mathcal{O}_P$.

We have that $\mathcal{D}_m$ first executes $\mathcal{D}_{m-1}$, then performs the oracle call $\mathcal{O}_P(\rho_s, \theta_{m-1})$, getting u_{m-1} and performs the computation of the next oracle call v_m (if $\mathcal{D}$ makes another oracle call), updates the history $\theta_m := (v_1, \ldots, v_m)$ and returns v_m if there is one or the output of $\mathcal{D}$ otherwise. $\mathcal{D}_m[\mathcal{A}_P^{\mathcal{O}}]$ first executes $\mathcal{D}_{m-1}[\mathcal{A}_P^{\mathcal{O}}]$, then performs the computation $\mathcal{A}_P^{\mathcal{O}}(\mathcal{M}_f, \rho_{r_1}, \theta'_{m-1})$ of u'_m, computes the next oracle call v'_m (if one is performed), updates $\theta'_m := (v'_1, \ldots, v'_m)$ and outputs either v_m of the output of $\mathcal{D}$.

We wish to use the definition of $\mathcal{O}$-simulation in order to conclude. However, we cannot directly use the $\mathcal{O}$-simulation, as $\mathcal{D}$ has access to an extra oracle $\mathcal{O}_Q$.

Part 1

We first prove that, assuming $\mathcal{A}_P^{\mathcal{O}}$ is a simulator of $\mathcal{O}_P$:

$$\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P}(\rho_r, 1^\eta) = c\} = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}_P^{\mathcal{O}}]^{\mathcal{O}}(\rho_r, 1^\eta) = c\}$$

This is a straightforward consequence of Lemma 9. Writing respectively $p_1^1(c) = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P}(\rho_r, 1^\eta) = c\}$ and $p_1^2(c) = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}_P^{\mathcal{O}}]^{\mathcal{O}}(\rho_r, 1^\eta) = c\}$, Using ρ_{r_1}, ρ_{r_2} as in Definition 5, we have

$$\begin{aligned}
p_1^1(c) &= \sum_{r_{\mathcal{B}}, r_2} \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P}(\rho_r, 1^\eta) = c \mid ([\![\overline{n}]\!]^\eta_{\rho_s}, \rho^{\mathcal{B}}_{\mathcal{O}}, \rho_{r_2}) = (\overline{v}, r_{\mathcal{B}}, r_2)\} \\
&\qquad \times \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{([\![\overline{n}]\!]^\eta_{\rho_s}, \rho^{\mathcal{B}}_{\mathcal{O}}, \rho_{r_2}) = (\overline{v}, r_{\mathcal{B}}, r_2)\} \\
p_1^2(c) &= \sum_{r_{\mathcal{B}}, r_2} \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}_P^{\mathcal{O}}]^{\mathcal{O}}(\rho_r, 1^\eta) = c| \ ([\![\overline{n}]\!]^\eta_{\rho_s}, \rho^{\mathcal{B}}_{\mathcal{O}}, \rho_{r_2}) = (\overline{v}, r_{\mathcal{B}}, r_2)\} \\
&\qquad \times \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{[\![\overline{n}]\!]^\eta_{\rho_s} = \overline{v}, \rho^{\mathcal{B}}_{\mathcal{O}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\}
\end{aligned}$$

We let

$$p_2^1(r_{\mathcal{B}}, r_2, \bar{v}, c) = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P}(\rho_r, 1^\eta) = c| \ ([\![\overline{n}]\!]^\eta_{\rho_s}, \rho^{\mathcal{B}}_{\mathcal{O}}, \rho_{r_2}) = (\overline{v}, r_{\mathcal{B}}, r_2)\}$$

and

$$p_2^2(r_{\mathcal{B}}, r_2, \bar{v}, c) = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}_P^{\mathcal{O}}]^{\mathcal{O}}(\rho_r, 1^\eta) = c| \ ([\![\overline{n}]\!]^\eta_{\rho_s}, \rho^{\mathcal{B}}_{\mathcal{O}}, \rho_{r_2}) = (\overline{v}, r_{\mathcal{B}}, r_2)\}$$

We use Definition 8 with $\mathcal{B}^{\mathcal{O}}(\rho_{r_2}, \eta, \phi)$ as the machine that simulates $\mathcal{D}_m$ for $m = |\phi|$ and using ϕ instead of querying the oracle. Let us define ϕ^i_m, θ^i_m for $i = 1, 2$ as in Definition 8. Note that with the definition of $\mathcal{D}$, $\mathcal{B}$ uses prefixes for oracle calls, disjoint from those used in $\mathcal{A}_P$, hence randomness used for oracle calls in $\mathcal{A}$ and $\mathcal{B}$ are disjoint. Let v^i_m be the last message of θ^i_m. By definition of $\mathcal{D}$ and $\mathcal{B}$ we have $v^1_m = v_m$ and $v^2_m = v'_m$. Choosing m such that $\mathcal{D}$ makes less than m oracle calls, we have

$$p_2^i(r_{\mathcal{B}}, r_2, \bar{v}, c) = \textstyle\sum_{\bar{x} \text{ s.t. } x_m = c, \bar{y}} \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta^i_m = \overline{x}, \phi^i_m = \overline{y}| \ ([\![\overline{n}]\!]^\eta_{\rho_s}, \rho^{\mathcal{B}}_{\mathcal{O}}, \rho_{r_2}) = (\overline{v}, r_{\mathcal{B}}, r_2)\}.$$

Lemma 9 yields for all $r_{\mathcal{B}}, r_2, c$ that $p_2^2(r_{\mathcal{B}}, r_2, c) = p_2^1(r_{\mathcal{B}}, r_2, c)$, which concludes part 1.

Part 2

We now prove that:

$$\begin{array}{c}
\forall \mathcal{D}.\ \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P}(\rho_r, 1^\eta) = c\} = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}_P^{\mathcal{O}}]^{\mathcal{O}}(\rho_r, 1^\eta) = c\} \\
\Rightarrow \\
\forall \mathcal{D}.\ \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P,\mathcal{O}_Q}(\rho_r, 1^\eta) = c\} = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}_P^{\mathcal{O}}]^{\mathcal{O},\mathcal{O}_Q}(\rho_r, 1^\eta) = c\}
\end{array} \tag{1}$$

We are thus going to show that, with the interpretation of $\overline{n}$ fixed, we can simulate $\mathcal{O}_Q$ in some $\mathcal{D}'$ by sampling in ρ_r instead of ρ_s. However, both computations of $\mathcal{O}_P$ and $\mathcal{O}_Q$ depend on ρ_s. This is where we need the assumptions that $\overline{n}$ contains the shared secrets between P and Q, as well as the splitting of ρ_r.

For any machine $\mathcal{M}^{\mathcal{O},\mathcal{O}_Q}$, we let $[\mathcal{M}]^{\mathcal{O}}_{\overline{n}}$ be the machine that executes $\mathcal{M}$, simulating $\mathcal{O}_Q$ for a fixed value $\overline{v}$ of $\overline{n}$. The machine samples the names appearing in Q and not in $\overline{n}$ and hard codes the interpretation of $\overline{n}$.

More precisely, we write $\mathcal{O}_Q(\rho_s, \theta) := \mathcal{O}_Q((\rho_{s_0}, \rho_{s_1}, \rho_{s_2}), \theta)$ where ρ_{s_0} is used for the sampling of $\overline{n}$, ρ_{s_1} for the sampling of other names in Q, and ρ_{s_2} for the reminder.

Then $[\mathcal{M}]^{\mathcal{O}}_{\overline{n}}(\rho_r, 1^\eta)$ is the machine that:

- Splits ρ_r into two infinite and disjoints ρ_{sQ}, ρ_{rM} and initializes an extra tape θ to zero.
- Simulates $\mathcal{M}(\rho_{rM}, 1^\eta)$ but every time $\mathcal{M}$ calls $\mathcal{O}_Q$ with input u, the machine adds u to θ, and produces the output of $\mathcal{O}_Q((\overline{v}, \rho_{rQ}, 0), \theta)$.

Such a machine runs in deterministic polynomial time (w.r.t. η). For any machine $\mathcal{M}^{\mathcal{O},\mathcal{O}_Q,\mathcal{O}_P}$, we similarly define $[\mathcal{M}]^{\mathcal{O},\mathcal{O}_P}_{\overline{n}}$. Now, we have that, for any c, by letting, for any X and U, $\mathbb{P}^{c,\overline{v}}_X(U) := \mathbb{P}_X\{U = c \mid [\![\overline{n}]\!]^\eta_{\rho_s} = \overline{v}\}$:

$$
\begin{array}{ll}
& \mathbb{P}^{c,\overline{v}}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}(\mathcal{D}^{\mathcal{O},\mathcal{O}_P(\rho_{s_0},\rho_{s_1},\rho_{s_2}),\mathcal{O}_Q(\rho_{s_0},\rho_{s_1},\rho_{s_2})}(\rho_r, 1^\eta)) \\
=^1 & \mathbb{P}^{c,\overline{v}}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}(\mathcal{D}^{\mathcal{O},\mathcal{O}_P(\rho_{s_0},\rho_{s_1},\rho_{s_2}),\mathcal{O}_Q(\rho_{s_0},\rho_{s_1},0)}(\rho_r, 1^\eta)) \\
=^2 & \mathbb{P}^{c,\overline{v}}_{\rho_{s_1},\rho_{s_2},\rho_r,\rho_{\mathcal{O}}}(\mathcal{D}^{\mathcal{O},\mathcal{O}_P(\rho_{s_0},0,\rho_{s_2}),\mathcal{O}_Q(\rho_{s_0},\rho_{s_1},0)}(\rho_r, 1^\eta)) \\
=^3 & \mathbb{P}^{c,\overline{v}}_{\rho_{s_1},\rho_{s_2},\rho_r,\rho_{\mathcal{O}}}(\mathcal{D}^{\mathcal{O},\mathcal{O}_P(\overline{v},0,\rho_{s_2}),\mathcal{O}_Q(\overline{v},\rho_{s_1},0)}(\rho_r, 1^\eta)) \\
=^4 & \mathbb{P}^{c,\overline{v}}_{\rho_{sQ},\rho_s,\rho_{rD},\rho_{\mathcal{O}}}(\mathcal{D}^{\mathcal{O},\mathcal{O}_P(\overline{v},0,\rho_s),\mathcal{O}_Q(\overline{v},\rho_{sQ},0)}(\rho_r, 1^\eta)) \\
=^5 & \mathbb{P}^{c,\overline{v}}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}([\mathcal{D}]^{\mathcal{O},\mathcal{O}_P(\rho_s)}_{\overline{n}}(\rho_r, 1^\eta)) \text{ (ii)}
\end{array}
$$

Since

1. $\mathcal{O}_Q$ does not access ρ_{s_2}
2. $\mathcal{O}_P$ does not access ρ_{s_1}
3. We are sampling under the assumption that $[\![\overline{n}]\!]^\eta_{\rho_s} = \overline{v}$, i.e., ρ_{s_0} is equal to $\overline{v}$.
4. Renaming of tapes
5. By construction

And we also have similarly that, for any c:

$$
\begin{array}{l}
\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}^{\mathcal{O}}_P]^{\mathcal{O},\mathcal{O}_Q}(\rho_r, 1^\eta) = c \mid [\![\overline{n}]\!]^\eta_{\rho_s} = \overline{v}\} \\
= \ \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{[\mathcal{D}[\mathcal{A}^{\mathcal{O}}_P]]^{\mathcal{O}}_{\overline{v}}(\rho_r, 1^\eta) = c \mid [\![\overline{n}]\!]^\eta_{\rho_s} = \overline{v}\} \text{ (iii)}
\end{array}
$$

By applying the left-handside of (1) to $[\mathcal{D}]^{\mathcal{O},\mathcal{O}_P(\rho_s)}_{\overline{n}}(\rho_r, 1^\eta)$ and $[\mathcal{D}[\mathcal{A}^{\mathcal{O}}_P]_j]^{\mathcal{O}}_{\overline{v}}(\rho_r, 1^\eta)$, and using (ii) and (iii), we can conclude by transitivity. We conclude the proof of the lemma by putting Part 1 and Part 2 together.

□

We now prove the converse direction.

Lemma 11. *Given an oracle $\mathcal{O}$ with support $\overline{n}$, a cryptographic library $\mathcal{M}^f$, protocols P, Q such that $\mathcal{N}(P) \cap \mathcal{N}(Q) \subset \overline{n}$, if there is a PTOM $\mathcal{A}_P^{\mathcal{Q}}$ such that, for every PTOM $\mathcal{D}^{\mathcal{O},\mathcal{O}_P,\mathcal{O}_Q}$, for every η, every $\overline{v} \in D_{\overline{n}}^{\eta}$ and every $c \in \{0,1\}^\star$,*

$$\begin{aligned}&\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P,\mathcal{O}_Q}(\rho_r, 1^\eta) = c \mid [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\} \\ &\quad = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}_P^{\mathcal{Q}}]^{\mathcal{O},\mathcal{O}_Q}(\rho_r, 1^\eta) = c \mid [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\}\end{aligned}$$

then $\nu\overline{n}.P$ is $\mathcal{O}$-simulatable.

Proof. Let $\mathcal{B}$ be a PTOM, η, an interpretation $\overline{v} \in D_{\overline{n}}^{\eta}$ and $m \in \mathcal{N}$, we must prove that the output distribution of B will be the same whether it interacts m-th time with $\mathcal{A}_P^{\mathcal{Q}}$ or $\mathcal{O}_P$. We define $\mathcal{D}$ as follows. For $i := 0$ to $m-1$, $\mathcal{D}$ computes $w_i := \mathcal{B}(\alpha_1, \ldots, \alpha_i)$. Then $\mathcal{D}$ calls $\mathcal{O}_P$ with w_i and let α_{i+1} be the reply. $\mathcal{D}$ finally outputs α_m. We denote by w_i' and α_i' the corresponding values for $\mathcal{D}[\mathcal{A}_P^{\mathcal{Q}}]^{\mathcal{O},\mathcal{O}_Q}$

Let us denote

$$\begin{aligned}\phi_{k+1}^2 &= \phi_k^2, \mathcal{O}_P(\rho_s, \theta_k^2) \\ \phi_{k+1}^1 &= \phi_k^1, \mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\mathcal{M}_f, \rho_{r_1}, \theta_k^1, \eta) \\ \theta_{k+1}^i &= \theta_k^i, \mathcal{B}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\mathcal{M}_f, \rho_{r_2}, \eta, \phi_k^i)\end{aligned}$$

for $0 \leq k < m$ and $\phi_0 = \theta_0 = \emptyset$.

We have by construction of $\mathcal{D}$ for any c:

$$\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{w_m = c \mid [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\} = \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{O}_P(\rho_s, \theta_m^2) = c \mid [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\}$$

and

$$\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{w_m' = c \mid [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\} = \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\mathcal{M}_f, \rho_{r_1}, \theta_m^1, \eta) = c \mid [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\}$$

The hypothesis gives us that :

$$\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{w_m = c \mid [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\} = \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{w_m' = c \mid [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\}$$

So we conclude that:

$$\begin{aligned}&\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\mathcal{M}_f, \rho_{r_1}, \theta_m^1, \eta) = c \mid [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\} \\ &\quad = \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{O}_P(\rho_s, \theta_m^2) = c \mid [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\}\end{aligned}$$

□

We can finally conclude, as Lemmas 10 and 11 directly yields that Definition 8 is equivalent to Definition 8 simply by taking Q as the empty protocol.

Example 3.1. We fix first $\mathcal{M}_f$ (in an arbitrary way). We consider the following handshake protocol, in which n, r, k, r' are names:

$$\begin{aligned}A :=\quad &\mathsf{in}\ (c_A, x_0).\mathtt{out}(c_A, \mathtt{enc}(n, r, k)).\ \mathsf{in}\ (c_A, x). \\ &\quad \mathsf{if}\ \mathtt{dec}(x, k) = \langle n, 1\rangle\ \mathsf{then}\ \mathtt{out}(c_A, \mathtt{ok}) \\ \|\quad B :=\quad &\mathsf{in}\ (c_B, y).\mathtt{out}(c_B, \mathtt{enc}(\langle \mathtt{dec}(y, k), 1\rangle, r', k))\end{aligned}$$

We consider the oracle $\mathcal{O}_k^{\mathtt{enc},\mathtt{dec}}$ that, when receiving $\langle t, m\rangle$ as input, answers $enc(m, r_o, k)$ if $t =$ `"enc"`, and $dec(m, l)$ if $t =$ `"dec"` (the oracle actually also expects an handle for the secret key and a tag to specify where to sample r_o). We can easily prove that $\nu k.A$ is

$\mathcal{O}_k^{\mathtt{enc},\mathtt{dec}}$-simulatable, as the attacker can sample an arbitrary n', use the oracle to compute $\mathtt{enc}(n', r_o, k)$ (which has the same distribution as $\mathtt{enc}(n', r, k)$ for any fixed value of k) with the request $\langle \mathtt{"enc"}, n\rangle$, and $\mathtt{dec}(x, k)$ with the request $\langle \mathtt{"dec"}, x\rangle$.

Intuitively, the shared secret k is only used in A in ways that are directly simulatable with the oracle, and A is thus $\mathcal{O}$-simulatable.

Thanks to the more intuitive Definition of simulatability (cf. Definition 8 for details), proving simulatability is in practice a syntactic verification. With $\mathcal{O}_k^{\mathtt{enc},\mathtt{dec}}$ from the previous example, $\nu k.P$ is $\mathcal{O}$-simulatable for any P where all occurrences of k occurs at key position, and all encryptions use fresh randoms.

Let us explain why the previous examples illustrate the need for prefixed models.

Example 3.2. We take a more formal view on Example 3.1.

Let $\mathcal{O}$ be the encryption-decryption oracle: it expects an input $\langle \mathtt{"dec"}, m\rangle$ or $\langle \mathtt{"enc"}, m\rangle$, a key $s = 1$ (only one encryption key is considered), an input tag t and a security parameter η and returns

- $\mathtt{enc}(m, r, k)$ if the query is prefixed by $\mathtt{"enc"}$, k is the secret value extracted from ρ_s corresponding to the key 1, r is drawn from $\rho_{\mathcal{O}}$ and associated with the tag t (via e_1).
- $\mathtt{dec}(m, k)$ if the query is prefixed by $\mathtt{"dec"}$, k is the secret value extracted from ρ_s corresponding to the key 1
- an error message otherwise (either the primitives fail or the query does not have the expected format).

The goal is to show that $\nu k.A$ is $\mathcal{O}$-simulatable. (So, here, B is useless, and we let P be A).

$\mathcal{O}_P$ is then defined as follows:

- On input w_1, with an empty history, it outputs $[\![\mathtt{enc}(n, r, k)]\!]^{\eta}_{\rho_s}$ and writes w_1 on the history tape.
- On input w_2 with a non empty history tape, it outputs $\mathtt{ok}$ if $[\![\mathtt{dec}(x, k)]\!]^{\eta, x\mapsto w_2}_{\rho_s} = [\![\langle n, 1\rangle]\!]^{\eta}_{\rho_s}$ and an error otherwise.

The machine $\mathcal{A}^{\mathcal{O}}(\rho_{r_1}, \theta, \eta)$ is then defined as follows:

- If $\theta = \{m_1\}$
  1. $\mathcal{A}$ draws α (for the value of n) from ρ_{r_1} and draws t from ρ_{r_1}
  2. calls $\mathcal{O}$ with $(\langle \mathtt{"enc"}, \alpha\rangle, 1, t)$ and gets back the bitstring $[\![\mathtt{enc}(n, r, z)]\!]^{\eta, z\mapsto [\![k]\!]^{\eta}_{\rho_s}}_{\rho_{r_1}, \rho_{\mathcal{O}}}$. The interpretation of k is indeed fixed at once since it belongs to the "shared" names bounded by ν.
  3. outputs $[\![\mathtt{enc}(x, r, z)]\!]^{\eta, x\mapsto\alpha, z\mapsto [\![k]\!]^{\eta}_{\rho_s}}_{\rho_{r_1}}$
- If $\theta = (m_1, m_2)$,
  1. calls $\mathcal{O}$ with $(\langle \mathtt{"dec"}, m_2\rangle, 1, -)$ and gets back the bitstring $w = [\![\mathtt{dec}(y, z)]\!]^{y\mapsto m_2, z\mapsto [\![k]\!]^{\eta}_{\rho_s}}$ or an error message.

2. checks whether $w = [\![\langle n, 1\rangle]\!]^{\eta}_{\rho_{r_1}}$. If it is the case, then outputs `ok`.

Now, consider an arbitrary PTOM $\mathcal{B}^{\mathcal{O}}$.

- $\phi_1^1 = [\![\mathtt{enc}(n, x, k)]\!]^{\eta, x \mapsto s_1}_{\rho_{r_1}}$ where s_1 is the randomness used by $\mathcal{O}$ when queried with $[\![t]\!]_{\rho_{r_1}}$ (note: we will see that it does matter to be very precise here; we cannot simply claim that the value of x is just a randomness drawn by $\mathcal{O}$).
- $\phi_1^2 = [\![\mathtt{enc}(n, r, k)]\!]^{\eta}_{\rho_s}$
- $\theta_i^1 = w_i$, an arbitrary bitstring, computed by $\mathcal{B}^{\mathcal{O}}$ using the oracle $\mathcal{O}$, ϕ_i^1 and the random tape ρ_{r_2}.
- $\phi_2^1 = \phi_1^1, \mathtt{ok}$ if
  $[\![\mathtt{dec}(y, z)]\!]^{y \mapsto w_1, z \mapsto [\![k]\!]^{\eta}_{\rho_s}} = [\![\langle n, 1\rangle]\!]^{\eta}_{\rho_{r_1}}$ and an error
  otherwise
- $\phi_2^2 = \phi_1^2, \mathtt{ok}$ if $[\![\mathtt{dec}(x, k)]\!]^{\eta, x \mapsto w_2}_{\rho_s} = [\![\langle n, 1\rangle]\!]^{\eta}_{\rho_s}$ and an error otherwise

$\mathcal{A}$ $\mathcal{O}$-simulates $\nu k.P$ iff, for every $v = [\![k]\!]_{\rho_s}$,

$$\begin{aligned}&\mathbb{P}_{\rho_s, \rho_{r1}, \rho_{r_2}, \rho_{\mathcal{O}}}\{[\![\mathtt{dec}(y, z)]\!]^{y \mapsto w_1, z \mapsto v} = [\![\langle n, 1\rangle]\!]^{\eta}_{\rho_{r_1}}\}\\ &\quad = \mathbb{P}_{\rho_s, \rho_{r_1}, \rho_{r2}, \rho_{\mathcal{O}}}\{[\![\mathtt{dec}(x, k)]\!]^{\eta, x \mapsto w_2}_{\rho_s} = [\![\langle n, 1\rangle]\!]^{\eta}_{\rho_s}\}\end{aligned}$$

First, the distributions of ϕ_1^1 and ϕ_1^2 are identical. ϕ_1^1 depends on ρ_{r_1} and $\rho_{\mathcal{O}}$, while ϕ_1^2 depends on ρ_s only. The distributions of $\phi_1^1, [\![\langle n, 1\rangle]\!]_{\rho_{r_1}}$ and $\phi_1^2, [\![\langle n, 1\rangle]\!]_{\rho_s}$ are also identical.

Now the distributions $w_1 = \mathcal{B}^{\mathcal{O}}(\phi_1^1, \rho_{r_2}), [\![\langle n, 1\rangle]\!]_{\rho_{r_1}}$ and $w_2 = \mathcal{B}^{\mathcal{O}}(\phi_2^1, \rho_{r_2}), [\![\langle n, 1\rangle]\!]_{\rho_s}$ are equal *if the randomness used by $\mathcal{B}$ are disjoint from the random coins used in ϕ_1^1, ϕ_1^2*. This is why there is an assumption that ρ_{r_1} and ρ_{r_2} are disjoint and why *it should be the case* that the random coins used in the oracle queries of $\mathcal{B}$ are distinct from the ones used in the oracle queries of $\mathcal{A}$. This can be ensured by the disjointness of tags used by $\mathcal{A}$ and $\mathcal{B}$ respectively.

With these assumptions, we get the identity of the distributions of $\mathtt{dec}(w_1, v), [\![\langle n, 1\rangle]\!]_{\rho_s}$ and $\mathtt{dec}(w_2, v), [\![\langle n, 1\rangle]\!]_{\rho_s}$, hence the desired result.

Without these assumptions (for instance non-disjointness of tags used by $\mathcal{B}$, $\mathcal{A}$), $\mathcal{B}$ can query $\mathcal{O}$ with a random input and a random tag, say n', t'. As above, we let s_1 be the random value drawn by $\mathcal{O}$ corresponding to the tag t'. Then $\mathbb{P}\{[\![n]\!]_{\rho_s} = n' \wedge [\![r]\!]_{\rho_s} = s_1\} = \frac{1}{2^{2\eta}}$ while

$$\begin{aligned}\mathbb{P}\{[\![n]\!]_{\rho_{r_1}} = n' \wedge [\![r]\!]_{\rho_{r_1}} = s_1\} &= \tfrac{1}{2^{\eta}}\mathbb{P}\{[\![t]\!]_{\rho_{r_1}} = [\![t']\!]_{\rho_{r_2}} \vee ([\![t]\!]_{\rho_{r_1}} \neq [\![t']\!]_{\rho_{r_2}} \wedge [\![r]\!]_{\rho_{r_1}} = [\![r']\!]_{\rho_{\mathcal{O}}})\}\\ &= \tfrac{1}{2^{\eta}} \times (\tfrac{1}{2^{\eta}} + \tfrac{2^{\eta}-1}{2^{\eta}} \times \tfrac{1}{2^{\eta}})\\ &= \tfrac{1}{2^{2\eta}}(2 - \tfrac{1}{2^{\eta}})\end{aligned}$$

In other words, the collision is more likely to occur since it can result from either a collision in the tags or a collision in the randomness corresponding to different tags.

As demonstrated in the previous example, it is necessary to assume that oracle randomness used by the simulator queries and the attacker queries are disjoint. The simplest way of ensuring this is to force all tags of oracle calls to be prefixed. We show here that this assumption can be made without loss of generality.

Definition 12. Given a PTOM $\mathcal{A}^{\mathcal{O}}$ and a constant c. We define $\mathcal{A}^{\mathcal{O}}_{pref-c}$ as a copy of $\mathcal{A}$, except that all calls to the oracle of the form $\overline{w}, r, s$ are replaced with calls of the form $\overline{w}, c\cdot r, s$, where the $\cdot$ denotes the concatenation of bitstrings.

The following lemma shows that we can, w.l.o.g., consider models, in which the tags are prefixed.

Lemma 13. *For any non-empty constant c and any PTOM $\mathcal{A}^{\mathcal{O}}$, we haves*

$$\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\rho_r,1^\eta)=1\} = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}_{pref-c}(\rho_r,1^\eta)=1\}$$

Proof. We fix a constant c, for any oracle $\mathcal{O}$ (with functions n, e_1, e_2), we define $\mathcal{O}_{pref-c}$ (with mapping function n', e_1', e_2') the copy of $\mathcal{O}$ such that:

$$n'(w,s,r) = n(w,s,c|r)$$

n is injective by definition, so n' is injective too. For any $v \in \{0,1\}^\eta$, as all extractions of e_1 are unique for each value of n and their length only depends on η, we have for any w, r, s

$$\mathbb{P}_{\rho_{\mathcal{O}}}\{e_1(n(w,s,r),\eta,\rho_{\mathcal{O}})=v\} = \mathbb{P}_{\rho_{\mathcal{O}}}\{e_1'(n'(w,s,r),\eta,\rho_{\mathcal{O}})=v\}$$

This implies that for any input, $\mathcal{O}$ and $\mathcal{O}_{pref-c}$ will produce the same output distribution. So $\mathcal{A}^{\mathcal{O}}$ and $\mathcal{A}^{\mathcal{O}_{pref-c}}$ will produce the same distributions for any input. We conclude by remarking that $\mathcal{A}^{\mathcal{O}_{pref-c}}$ and $\mathcal{A}^{\mathcal{O}}_{pref-c}$ behaves the same by construction. □

An immediate consequence of this Lemma is that for all indistinguishability results, we can, w.l.o.g., constrain attackers to only use prefixed oracle calls.

In particular it implies equivalence between indistinguishability in a computational model and indistinguishability for prefixed distinguishers in the prefixed computational model.

Thanks to the previous Definitions, simulatability is stable under composition operators. This is an important feature of the notion of simulatability, as it allows to reduce the simulation of large processes to the simulation of simpler processes.

Theorem 1. *Given an oracle $\mathcal{O}$, protocols P, Q, and $\overline{n} = \mathcal{N}(P) \cap \mathcal{N}(Q)$, if*

- *$\nu\overline{n}.P$ is $\mathcal{O}$-simulatable*
- *$\nu\overline{n}.Q$ is $\mathcal{O}$-simulatable*

Then $\nu\overline{n}.P \| Q$ and $\nu\overline{n}.P; Q$ are $\mathcal{O}$-simulatable.

Proof. Let $\mathcal{D}$ be an arbitrary PTOM. By Lemma 10, there is a machine $\mathcal{A}^{\mathcal{O}}_P$ s.t.

$$\begin{aligned}&\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P,\mathcal{O}_Q}(\rho_r,1^\eta)=c \mid [\![\overline{n}]\!]^\eta_{\rho_s}=\overline{v}\}\\ &\quad=\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}^{\mathcal{O}}_P]^{\mathcal{O},\mathcal{O}_Q}(\rho_r,1^\eta)=c \mid [\![\overline{n}]\!]^\eta_{\rho_s}=\overline{v}\}\end{aligned}$$

Applying once more the Lemma 10, there is a machine $\mathcal{A}^{\mathcal{O}}_Q$ s. t., for every $c \in \{0,1\}^\star$,

$$\begin{aligned}&\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}^{\mathcal{O},\mathcal{O}_P,\mathcal{O}_Q}(\rho_r,1^\eta)=c \mid [\![\overline{n}]\!]^\eta_{\rho_s}=\overline{v}\}\\ &\quad=\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{D}[\mathcal{A}^{\mathcal{O}}_P][\mathcal{A}^{\mathcal{O}}_Q]^{\mathcal{O}}(\rho_r,1^\eta)=c \mid [\![\overline{n}]\!]^\eta_{\rho_s}=\overline{v}\}\end{aligned}$$

We define $\mathcal{A}^{\mathcal{O}}_{P\|Q}(\mathcal{M}^f, \rho_{r_1}, \theta, 1^\eta, m)$ as the machine that behaves as $\mathcal{A}^{\mathcal{O}}_P(\mathcal{M}^f, \rho_{r_1,P}, \theta_P, 1^\eta, m)$ (resp. $\mathcal{A}^{\mathcal{O}}_Q(\mathcal{M}^f, \rho_{r_1,Q}, \theta_Q, m)$) if m is a message supposed to be handled by P (resp. by Q) (use of action determinism) Then the result is appended to θ_P (resp. θ_Q). This assumes (this is an invariant) that θ can be split into θ_P and θ_Q.

We note that $\mathcal{D}[\mathcal{A}^{\mathcal{O}}_P][\mathcal{A}^{\mathcal{O}}_Q]^{\mathcal{O}} = \mathcal{D}[\mathcal{A}^{\mathcal{O}}_{P\|Q}]^{\mathcal{O}}$. Then we use Lemma 11 to conclude. □

Alternative notions of simulatability We discuss here some variation on our notion of simulatability. First, let us note that our notion of simulatability assumes that models are prefixed. As demonstrated previously this is necessary in order to get an achievable notion of simulatability. We will therefore not consider models that are not prefixed. We may consider variants of simulatability, depending on the order of the quantifiers and sharing of randomness between simulator and distinguisher. We define simulatability as the existence of a simulator that works for all distinguishers. In other words our ordering of quantifier is:

$$\exists \mathcal{A}^{\mathcal{O}}(\rho_{r_1}) \forall \mathcal{D}(\rho_{r_2})$$

In a prefixed model, we believe that switching the quantifiers lead to the same notion:

$$\exists \mathcal{A}^{\mathcal{O}}(\rho_{r_1}) \forall \mathcal{D}(\rho_{r_2}) \Leftrightarrow \forall \mathcal{D}(\rho_{r_2}) \exists \mathcal{A}^{\mathcal{O}}(\rho_{r_1})$$

We provide no proof, but the intuition is that there exists a "universal" distinguisher, namely the PTOM $\mathcal{D}$, which performs any possible queries with uniform probability. Now, considering any other distinguisher $\mathcal{D}'$, as the simulator $\mathcal{A}^{\mathcal{O}}$ for $\mathcal{D}$ has to provide the exact same distribution as the protocol for each query of $\mathcal{D}$, as $\mathcal{D}$ performs all possible queries (with very small probability), $\mathcal{A}^{\mathcal{O}}$ will also be a correct simulator for $\mathcal{D}'$.

Another alternative is to allow the simulator and the distinguisher to share the same randomness. Then, $\exists \mathcal{A}^{\mathcal{O}}(\rho_r) \forall \mathcal{D}(\rho_r)$ seems to provide an unachievable definition. Indeed, if the simulator is not allowed to use private randomness while the protocol is, the simulator cannot mimic the probabilistic behavior of the protocol.

The last possibility however seems to offer an alternative definition for simulatability:

$$\forall \mathcal{D}(\rho_r) \exists \mathcal{A}^{\mathcal{O}}(\rho_r)$$

This seems to be a weaker definition than ours as the choices of the simulator can depend on the ones of the distinguisher. It may simplify (slightly) the proofs for the main theorem, but it would create issues for the unbounded replication as it would break uniformity of reductions (since the runtime of the simulator may now depend on the environment it is running in).

3.2 Generic Oracles for Tagged Protocols

In order for our definition of simulatability to be useful, the design of oracles is a key point. They need to be:

1. generic/simple, yet powerful enough so that protocols can be easily shown to be simulatable,

2. restrictive enough so that proving protocols in the presence of oracles is doable.

We provide here with examples of such oracles, namely generic tagged oracles for signature, that will be parameterized by arbitrary functions, together with security properties that are still true in the presence of tagged oracles.

In practice, protocols that use some shared secrets use tags, for instance string prefixes, to ensure that messages meant for one of the protocol cannot be confused with messages meant for the other one. These tags can ensure what is called "domain separation" of the two protocols, ensuring that the messages obtained from one cannot interfere with the security of the second protocol. These tags can be explicit, for instance by adding a fixed constant to

the messages, or implicit, where each message of a protocol depend on some fresh randomness that can be used to define some kind of session identifier.

We define generic oracles for decryption and signatures, parameterized by an abstract tagging function T and a secret key sk, that allow to perform a cryptographic operation with the key sk, on any message m satisfying $T(m)$. T can then simply check the presence of a prefix, or realize some implicit tagging, checking that the message depends on the randomness used by a specific session.

After defining those generic oracles, we define generic axioms, parameterized by T, that allow to perform proofs against attackers with access to the oracle. The generic axiom for signatures (or any other primitive) are implied by the classical cryptographic axioms.

We see tagging as a boolean function T computable in polynomial time over the interpretation of messages. For instance, if the messages of protocol P are all prefixed with the identifier id_P, T is expressed as $T(m) := \exists x.m = \langle id_P, x\rangle$. In a real life protocol, id_P could for instance contain the name and version of the protocol.

Intuitively tagged oracles produce the signature of any properly tagged message and allow to simulate P.

With these oracles, an immediate consequence of the composition Theorems found in Section 4 is the classical result that if two protocols tag their messages differently, they can be safely composed [28]. Note that as our tag checking function is an arbitrary boolean function: tagging can be implicit, as illustrated in our applications in Section 6.

As an example, we provide two oracles, one for encryption and one for signing, that allow to simulate any protocol that only produces messages that are well tagged for T.

Definition 14. Given a name sk and a tagging function T, we define the generic signing oracle $\mathcal{O}^{sign}_{T,sk}$ and the generic decryption oracle $\mathcal{O}^{dec}_{T,sk}$ as follows:

$$\begin{aligned}
\mathcal{O}^{\mathtt{sign}}_{T,sk}(m) := \quad & \mathsf{if}\ T(m)\ \mathsf{then} \\
& \quad \mathrm{output}(\mathtt{sign}(m, sk)) \\
\mathcal{O}^{\mathtt{dec}}_{T,sk}(m) := \quad & \mathsf{if}\ T(\mathtt{dec}(m, sk))\ \mathsf{then} \\
& \quad \mathrm{output}(\mathtt{dec}(m, sk))
\end{aligned}$$

Any well-tagged protocol according to T, i.e., a protocol that only decrypts or signs well tagged messages, will be simulatable using the previous oracles. Hence we meet the goal 1 stated at the beginning of this section, as this can be checked syntactically on a protocol. We provide, as an example, the conditions for a tagged signature.

Example 3.3. Any protocol P whose signatures are all of the form $\mathsf{if}\ T(t)\ \mathsf{then}\ \mathtt{sign}(t, sk)$ for some term t (that does not use sk) is immediately $\nu sk.P$ $\mathcal{O}^{\mathtt{sign}}_{T,sk}$-simulatable. Indeed, informally, all internal values of the protocol except sk can be picked by the simulator from its own randomness, while all terms using sk can be obtained by calls to the tagged signing oracle, as all signed terms in P are correctly tagged. Let us emphasize that the simulation holds for any specific value of sk, as the distribution of outputs is the same, whether it is the simulator that draws the internal names of P, except sk, or P itself.

As we need to perform cryptographic proofs in the presence of oracles, it is useful to define security properties that cannot be broken by attackers with access to these oracles (without having to consider the specific calls made to these oracles). The games defining these properties slightly differ from the classical security games. Consider the example of

signatures and the usual EUF-CMA game. If the attacker is, in addition, equipped with an oracle $\mathcal{O}$ that signs tagged messages, they immediately win the EUF-CMA game, "forging" a signature by a simple call to $\mathcal{O}$. We thus define a tagged unforgeability game (T-EUF-CMA), derived from the EUF-CMA game, where the adversary wins the game only if they are able to produce the signature of a message that is not tagged.

Definition 15. A signature scheme $(\mathsf{Sign}, \mathsf{Vrfy})$ is T-EUF-CMA secure for oracle $\mathcal{O}$ and interpretation of keys $\mathcal{A}_{sk}$ if, for any PTOM $\mathcal{A}$, the game described in Figure 3 returns `true` with probability (over $\rho_r, \rho_s, \rho_{\mathcal{O}}$) negligible in η.

Game $\mathrm{EUF\text{-}CMA}^{\Sigma,\mathcal{A}}_{T,sk}(\eta, \rho_r, \rho_s, \rho_{\mathcal{O}})$:	**Oracle** $\mathsf{Sign}(\mathsf{m})$:
$\mathsf{List} \leftarrow []$	$\mathsf{List} \leftarrow (\mathsf{m} : \mathsf{List})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow (\llbracket pk \rrbracket_{\rho_s}, \llbracket sk \rrbracket_{\rho_s})$	$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{m})$
$(\mathsf{m}, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}}),\mathsf{Sign}}(\mathsf{pk}, \eta, \rho_r)$	Return σ
Return $\neg T(\mathsf{m}) \wedge \mathsf{Vrfy}(\mathsf{pk}, \mathsf{m}, \sigma) \wedge \mathsf{m} \notin \mathsf{List}$	

Figure 3: Game for Tagged Unforgeability (T-EUF-CMA)

The main goal of the previous definition is to allow us to prove protocols in the presence of oracles (hence composed with simulated ones), reaching the goal 2 stated at the beginning of the section.

More precisely, one can, for instance, simply design a classical game based proof, reducing the security of the protocol to the security of the T-EUF-CMA game rather than the classical EUF-CMA game. This reasoning is valid as EUF-CMA implies T-EUF-CMA even in the presence of the corresponding oracle.

Proposition 16. *If a signature scheme* $(\mathsf{Sign}, \mathsf{Vrfy})$ *is* EUF-CMA *secure for keys given by* $\mathcal{A}_{sk}$*, then* $(\mathsf{Sign}, \mathsf{Vrfy})$ *is* T-EUF-CMA *secure for the oracle* $\mathcal{O}^{\mathtt{sign}}_{T,sk}$ *and the interpretation of keys* $\mathcal{A}_{sk}$*.*

Remark that the base assumptions made about the cryptographic primitives are classical ones, and thus the final proof of the composed protocol only depends on some classical cryptographic hypotheses.

4 Main Composition Theorems

We distinguish between two complementary cases. First, Theorem 2 covers protocols composed in a way where they do not share states besides the shared secrets (e.g., parallel composition of different protocols using the same master secret key). Second, Theorem 4 covers protocols passing states from one to the other (e.g., a key exchange passing an ephemeral key to a secure channel protocol). We finally extend these composition results to self-composition, i.e., proving the security of multiple sessions from the security of a single one or the security of a protocol lopping on itself, for instance a key renewal protocol.

4.1 Composition without State Passing

Essentially, if two protocols P, Q are indistinguishable, they are still indistinguishable when running in any simulatable context. The context must be simulatable for any fixed values

of the shared names of P, Q and the context. The context can contain parallel or sequential composition as illustrated by the following example.

Example 4.1. Let P, Q, R, S be protocols and $\mathcal{O}$ an oracle. Let $\overline{n} = \mathcal{N}(P\|Q) \cap \mathcal{N}(R\|S)$. If $P \cong_{\mathcal{O}} Q$ and $\nu\overline{n}.R\|S$ is $\mathcal{O}$-simulatable, then some applications of Theorem 2 can yield

1. $P\|R \cong_{\mathcal{O}} Q\|R$
2. $R; P \cong_{\mathcal{O}} R; Q$
3. $(R; P)\|S \cong_{\mathcal{O}} (R; Q)\|S$

We generalize the previous example to any simulatable context and to n protocols. For any integer n, we denote by $C[\__1, \ldots, \__n]$ a *context*, i.e., a protocol built using the syntax of protocols and distinct symbols $\__i$, viewed as elementary processes. $C[P_1, \ldots, P_n]$ is the protocol in which each hole $\_i$ is replaced with P_i.

Example 4.2. In the three examples of Example 4.1, in order to apply the next theorem, we respectively use as contexts

- $C[\__1] := \__1\|R$
- $C[\__1] := R; \__1$
- $C[\__1] := (R; \__1)\|S$.

In this first Theorem, no values (e.g., ephemeral keys) are passed from the context to the protocols. In particular, the protocols do not have free variables which may be bound by the context.

Theorem 2. *Given a cryptographic library $\mathcal{M}_f$ and an oracle $\mathcal{O}$, let $P_1, \ldots, P_n$, $Q_1, \ldots, Q_n$ be protocols and $C[\__1, \ldots, \__n]$ be a context such that all their channels are disjoint,* 0 *some constant, $\overline{n}$ a sequence of names and $c_1, \ldots, c_n$ fresh channel names. If*

1. $\mathcal{N}(C) \cap \mathcal{N}(P_1, \ldots, P_n, Q_1, \ldots, Q_n) \subseteq \overline{n}$
2. $\nu\overline{n}.C[\mathtt{out}(c_1, 0), \ldots, \mathtt{out}(c_n, 0)]$ *is $\mathcal{O}$-simulatable*
3. $P_1\| \ldots \|P_n \cong_{\mathcal{O}} Q_1\| \ldots \|Q_n$

Then

$$C[P_1, \ldots, P_n] \cong_{\mathcal{O}} C[Q_1, \ldots, Q_n]$$

Specifically[3]*, there exists a polynomial p_S (independent of C) such that, if p_C is the polynomial bound on the runtime of the simulator for C, we have,*

$$\mathrm{Adv}^{C[P_1,\ldots,P_n] \cong_{\mathcal{O}} C[Q_1,\ldots,Q_n]}(t)$$

$$\leq \mathrm{Adv}^{P_1\|\ldots\|P_n \cong_{\mathcal{O}} Q_1\|\ldots\|Q_n}\Big(p_S\big(t, n, |C|, p_C(t)\big)\Big)$$

[3]We provide, in this Theorem and the following ones, explicit advantages, as our constructions do not directly allow for unbounded replication. This will later be used to ensure that the advantage of the adversary only grows polynomially with respect to the number of sessions.

Note that the bound we obtain for the reduction is polynomial in the running time of the context. We denote by $\overline{C}$ the protocol C in which each $\_i$ is replaced with $\mathtt{out}(c_i, 0).\mathbf{0}$, where c_i is a channel name and 0 is a public value. Intuitively, $\overline{C}$ abstracts out the components P_i, only revealing which P_i is running at any time. The intuition behind the proof of the Theorem is then as follows. First, we show that $\overline{C}\|P_1\| \dots \|P_n \cong_{\mathcal{O}} \overline{C}\|Q_1\| \dots \|Q_n$ implies $C[P_1, \dots, P_n] \cong_{\mathcal{O}} C[Q_1, \dots, Q_n]$. This is done by a reduction, where we mainly have to handle the scheduling, which is possible thanks to the information leaked by $\overline{C}$, and the action determinism of the protocols. In a sense, this means that indistinguishability for protocols in parallel implies indistinguishability for any scheduling of those protocols. Secondly, by simulating $\overline{C}$ thanks to Proposition 7, the two hypothesis of the Theorem imply $\overline{C}\|P_1\| \dots \|P_n \cong_{\mathcal{O}} \overline{C}\|Q_1\| \dots \|Q_n$. The second part is where our notion of simulatability comes into play, and where it is essential to deal carefully with the shared secrets.

For our latter results, we must actually generalize slightly this Theorem. A use case is for instance when we want to prove that $P\|Q \cong P\|P$ implies that $\mathbf{if}\, b\, \mathbf{then}\, P\, \mathbf{else}\, Q \cong P$ for some boolean condition b. In this case, we actually need to rename the channels used by P and Q in the second protocol, so that both P and Q uses the same channels. We thus introduce a renaming on channels σ that allows us to compose components in an arbitrary way.

The generalized version of the Theorem is as follows.

Theorem 3. *Let $C[\_1, \dots, \_n]$ be a context. Let $P_1, \dots, P_n$, $Q_1, \dots, Q_n$ be protocols, and let $\sigma : \mathcal{C}(P_1, \dots, P_n) \mapsto \mathcal{C}$ such that $\overline{C}\|P_1\| \dots \|P_n$, $\overline{C}\|Q_1\| \dots \|Q_n$, $C[P_1\sigma, \dots, P_n\sigma]$, $C[Q_1\sigma, \dots, Q_n\sigma]$ are protocols. Given a cryptographic library $\mathcal{M}_f$, an oracle $\mathcal{O}$, if*

1. $\overline{n} \supseteq \mathcal{N}(C) \cap \mathcal{N}(P_1, \dots, P_n, Q_1, \dots, Q_n)$
2. *$\nu\overline{n}.\overline{C}$ is $\mathcal{O}$-simulatable*
3. $P_1\| \dots \|P_n \cong_{\mathcal{O}} Q_1\| \dots \|Q_n$

Then

$$C[P_1\sigma, \dots, P_n\sigma] \cong_{\mathcal{O}} C[Q_1\sigma, \dots, Q_n\sigma]$$

Specifically, there exists a polynomial p_S (independent of C) such that, if p_C is the polynomial bound on the runtime of the simulator for $\overline{C}$, we have,

$$\mathrm{Adv}^{C[P_1\sigma,\dots,P_n\sigma] \cong_{\mathcal{O}} C[Q_1\sigma,\dots,Q_n\sigma]}(t) \leq \mathrm{Adv}^{P_1\|\dots\|P_n \cong_{\mathcal{O}} Q_1\|\dots\|Q_n}\Big(p_S\big(t, n, |C|, |\sigma|, p_C(t)\big)\Big)$$

Proof. Let $\mathcal{A}$ be an attacker against

$$C[P_1\sigma, \dots, P_n\sigma] \cong_{\mathcal{O}} C[Q_1\sigma, \dots, Q_n\sigma].$$

In the scheduling part, we first build an attacker against

$$\overline{C}\|P_1\| \dots \|P_n \cong_{\mathcal{O}} \overline{C}\|Q_1\| \dots \|Q_n.$$

We then remove the context $\overline{C}$ through the $\mathcal{O}$-simulatability.

Scheduling part Let us construct $\mathcal{B}^{\mathcal{O},\mathcal{O}_{\overline{C}},\mathcal{O}_{R_1},\ldots,\mathcal{O}_{R_n}}$ with either for every i, $R_i = P_i$, or, for every i, $R_i = Q_i$. $\mathcal{B}^{\mathcal{O},\mathcal{O}_{\overline{C}},\mathcal{O}_{R_1},\ldots,\mathcal{O}_{R_n}}$ initially sets variables $c_1, \ldots, c_n$ to 0 (intuitively, c_i records which processes have been triggered) and sets $\overline{x}$ to the empty list. It then simulates $\mathcal{A}^{\mathcal{O},\mathcal{O}_{C[R_1\sigma,\ldots,R_n\sigma]}}$ but, each interaction with $\mathcal{O}_{C[R_1\sigma,\ldots,R_n\sigma]}$ and the corresponding request (c, m) is replaced with:

- if there exist i such that $c_i = 1$ and $c \in \mathcal{C}(R_i\sigma)$ then
  - query $\mathcal{O}_{R_i}$ with $(c\sigma^{-1}, m)$
  - if $\mathcal{O}_{R_i}$ returns $\bot$, then, if contexts C_1 and C_2 are such that $C[\__1, \ldots, \__n] = C_1[\__i; C_2]$, it adds to $\overline{x}$ the channels $\mathcal{C}(C_2)$. (This corresponds to the semantics of sequential composition: an error message disables the continuation).
  - else the answer (c', m') is changed $(c'\sigma, m')$ (and the simulation goes on)
- else if $c \in \mathcal{C}(\overline{C})$ and $c \notin \overline{x}$ then
  - query $\mathcal{O}_{\overline{C}}$ with (c, m)
  - if $\mathcal{O}_{\overline{C}}$ answers $\top$ on channel γ_i, set $c_i = 1$
  - else continue with the reply of $\mathcal{O}_{\overline{C}}$

This new attacker is basically simply handling the scheduling of the protocols, using the signals raised in the context to synchronize everything. The condition that there exists i such that $c_i = 1$ and $c \in \mathcal{C}(R_i)$ is always satisfied by a unique i, otherwise $C[P_1\sigma, \ldots, P_n\sigma]$ or $C[Q_1\sigma, \ldots, Q_n\sigma]$ would not be well formed.

The execution time of $\mathcal{B}$ then only depends on the number of channels in C, the size of the channel substitution σ, the number of protocols n in addition to the cost of simulating $\mathcal{A}$. Hence if t is the runtime of $\mathcal{A}$, there exists p_{S_1} such that the runtime of $\mathcal{B}$ is bounded (uniformly in C, $P_1, \ldots, P_n, Q_1, \ldots Q_n$) by $p_{S_1}(n, t, |C|, |\sigma|)$:

$$\mathrm{Adv}^{C[P_1,\ldots,P_n]\cong C[Q_1,\ldots,Q_n]}_{\mathcal{A}^{\mathcal{O}}}(t) \leq \mathrm{Adv}^{P_1\|\ldots\|P_n\cong Q_1\|\ldots\|Q_n}_{\mathcal{B}^{\mathcal{O},\mathcal{O}_{\overline{C}}}}(p_{S_1}(t, n, |C|, |\sigma|))$$

Simulatability Now, with the fact that $\nu\overline{n}.\overline{C}$ is $\mathcal{O}$-simulatable, we have a simulator $\mathcal{A}^{\mathcal{O}}_{\overline{C}}$ such that, thanks to Lemma 10, $\mathcal{B}[\mathcal{A}^{\mathcal{O}}_{\overline{C}}]^{\mathcal{O},\mathcal{O}_R}$ behaves exactly as $\mathcal{B}^{\mathcal{O},\mathcal{O}_{\overline{C}},\mathcal{O}_R}$. We have, for p_C the polynomial bound on the runtime of $\mathcal{A}_{\overline{C}}$, by Definition 5,

$$\mathrm{Adv}^{P_1\|\ldots\|P_n\cong Q_1\|\ldots\|Q_n}_{\mathcal{B}^{\mathcal{O},\mathcal{O}_{\overline{C}}}}(t) \leq \mathrm{Adv}^{P_1\|\ldots\|P_n\cong Q_1\|\ldots\|Q_n}_{\mathcal{B}[\mathcal{A}^{\mathcal{O}}_{\overline{C}}]^{\mathcal{O}}}(q(p_C(t) + t))$$

and finally,

$$\begin{aligned}&\mathrm{Adv}^{C[P_1\sigma,\ldots,P_n\sigma]\cong C[Q_1\sigma,\ldots,Q_n\sigma]}_{\mathcal{A}^{\mathcal{O}}}(t)\\ &\leq \mathrm{Adv}^{P_1\|\ldots\|P_n\cong Q_1\|\ldots\|Q_n}_{\mathcal{B}[\mathcal{A}^{\mathcal{O}}_{\overline{C}}]^{\mathcal{O}}}(q(p_C \circ p_{S_1}(n, t, |C|, |\sigma|) + p_{S_1}(n, t, |C|, |\sigma|)))\end{aligned}$$

□

Given a protocol P and a context C, for Theorem 2 to be used, we need an oracle such that:

1. the context C is simulatable with the oracle $\mathcal{O}$,

2. the protocol P is secure even for an attacker with access to $\mathcal{O}$ ($P \cong_{\mathcal{O}} Q$).

Our goal is to find an oracle that is generic enough to allow for a simple proof of indistinguishability of P and Q under the oracle, but still allows to simulate $\overline{C}$. Notably, if we take as oracle the protocol oracle corresponding to the context itself, we can trivially apply Theorem 2 but proving $P \cong_{\mathcal{O}} Q$ amounts to proving $C[P] \cong C[Q]$.

Application to tagged protocols We consider two versions of SSH, calling them SSH_2 and SSH_1, assuming that all messages are prefixed respectively with the strings "SSHv2.0" and "SSHv1.0". Both versions are using the same long term secret key sk for signatures. We assume that both versions check the string prefix.

To prove the security of SSH_2 running in the context of SSH_1, we can use Theorem 2. If we denote by I the idealized version of SSH_2, the desired conclusion is $SSH_2 \| SSH_1 \cong I \| SSH_1$. Letting $C[\__1] = \__1 \| SSH_1$, it is then sufficient to find an oracle $\mathcal{O}$ such that:

1. $\nu sk.SSH_1$ is $\mathcal{O}$-simulatable (the simulatability of C directly follows),
2. $SSH_2 \cong_{\mathcal{O}} I$

If we define the tagging function T_{SSH_1} that checks the prefix, SSH_1 is trivially $\mathcal{O}^{\mathtt{sign}}_{T_{SSH_1},sk}$-simulatable (see Definition 14) as SSH_1 does enforce the tagging checks. We thus let $\mathcal{O}$ be $\mathcal{O}^{\mathtt{sign}}_{T_{SSH_1},sk}$.

Assuming that `sign` verifies the classical EUF-CMA axiom, by Proposition 16, it also verifies the tagged version EUF-CMA$_{T_{SSH_1},sk}$. To conclude, it is then sufficient to prove that $SSH_2 \cong_{\mathcal{O}} I$ with a reduction to EUF-CMA$_{T_{SSH_1},sk}$.

Application to encrypt and sign For performances considerations, keys are sometimes used both for signing and encryption, for instance in the EMV protocol. In [29], an encryption scheme is proven to be secure even in the presence of a signing oracle using the same key. Our Theorem formalizes the underlying intuition, i.e. if a protocol can be proven secure while using this encryption scheme, it will be secure in any context where signatures with the same key are also performed.

4.2 Composition with State Passing

In some cases, a context passes a sequence of terms to another protocol. If the sequence of terms is indistinguishable from another one, we would like the two experiments, with either sequences of terms, to be indistinguishable.

Example 4.3. Let us consider the protocol $P(x_1, x_2) := \mathtt{in}(c, x).\mathtt{out}(c, \mathtt{enc}(x, x_1, x_2))$. We assume that we have a function kdf, which, given a random input, generates a suitable key for the encryption scheme. Let a random name $seed$ and let $C[\__1] := \mathsf{let}\ sk = \mathsf{kdf}(seed)\ \mathsf{in}\ \__1$. $C[\|^i P(r_i, sk)]$ provides an access to an encryption oracle for the key generated in C:

$$C[\|^i P(r_i, sk)] := \begin{array}{l} \mathsf{let}\ sk = \mathsf{kdf}(seed)\ \mathsf{in} \\ \qquad \|^i(\mathtt{in}(c, x).\mathtt{out}(c, \mathtt{enc}(x, r_i, sk))) \end{array}$$

A classical example is a key exchange, used to establish a secure channel. The situation is dual with respect to the previous theorem: contexts must be indistinguishable and the continuation must be simulatable.

Theorem 4. *Let C, C' be n-ary contexts such that each hole is terminal. Let $P_1(\overline{x}), \ldots, P_n(\overline{x})$ be parameterized protocols, such that channel sets are pairwise disjoint. Given a cryptographic library $\mathcal{M}_f$, an oracle $\mathcal{O}$, $\overline{n} \supseteq \mathcal{N}(C) \cap \mathcal{N}(P_1, \ldots, P_n)$, $\overline{t_1}, \ldots, \overline{t_n}, \overline{t'_1}, \ldots, \overline{t'_n}$ sequences of terms, $\widetilde{C} := C[\mathtt{out}(c_1, \overline{t_1}), \ldots, \mathtt{out}(c_n, \overline{t_n})]$ and $\widetilde{C'} := C'[\mathtt{out}(c_1, \overline{t'_1}), \ldots, \mathtt{out}(c_n, \overline{t'_n})]$. If $\widetilde{C} \| \mathtt{in}(c_1, \overline{x}).P_1(\overline{x}) \| \ldots \| \mathtt{in}(c_n, \overline{x}).P_n(\overline{x})$ is a protocol and:*

1. $\widetilde{C} \cong_{\mathcal{O}} \widetilde{C'}$
2. $\nu \overline{n}.\mathtt{in}(c_1, \overline{x}).P_1(\overline{x}) \| \ldots \| \mathtt{in}(c_n, \overline{x}).P_n(\overline{x})$ *is $\mathcal{O}$-simulatable*

then $C[P_1(\overline{t_1}), \ldots, P_n(\overline{t_n})] \cong_{\mathcal{O}} C'[P_1(\overline{t'_1}), \ldots, P_n(\overline{t'_n})]$

Specifically, there exists a polynomial p_S (independent of $P_1, \ldots, P_n$) such that if p_O is the polynomial bound on the runtime of the simulator for $P := \mathtt{in}(c_1, \overline{x}).P_1(\overline{x}) \| \ldots \| \mathtt{in}(c_n, \overline{x}).P_n(\overline{x})$, we have,

$$\mathrm{Adv}^{\,C[P_1(\overline{t_1}),\ldots,P_n(\overline{t_n})] \cong_{\mathcal{O}} C[P_1(\overline{t'_1}),\ldots,P_n(\overline{t'_n})]}(t) \leq \mathrm{Adv}^{\widetilde{C} \cong_{\mathcal{O}} \widetilde{C'}}\Big(p_S\big(t, n, |P|, p_P(t)\big)\Big)$$

$\widetilde{C}$ is the context, in which all the bound values (for instance the key derived by a key exchange) are outputted on distinct channels. $\widetilde{C'}$ corresponds to the idealized version. We can pass those bound values to another protocol P, if this protocol P can be simulated for any possible value of the bound values.

Proof. The proof is very similar to Theorem 2.

Let us assume that we have an attacker such that

$$\mathrm{Adv}\left(\mathcal{A}^{\mathcal{O}, \mathcal{O}_{C[P_1(\overline{t_1}),\ldots,P_n(\overline{t_n})]?C[P_1(\overline{t'_1}),\ldots,P_n(\overline{t'_n})]}}\right) = \epsilon_0$$

We denote $C_1 = C[\mathtt{out}(c_1, \overline{t_1}), \ldots, \mathtt{out}(c_n, \overline{t_n})]$, $C_2 = C[\mathtt{out}(c_1, \overline{t'_1}), \ldots, \mathtt{out}(c_n, \overline{t'_n})]$, $P'_1 = in(1, \overline{x}).P_1(\overline{x})$, $\ldots$, $P'_n = in(n, \overline{x}).P_n(\overline{x})$. We first construct an attacker against:

$$C_1 \| P'_1 \| \ldots \| P'_n \cong C_2 \| P'_1 \| \ldots \| P'_n$$

Let us consider $\mathcal{B}^{\mathcal{O}, \mathcal{O}_D, \mathcal{O}_{P'_1}, \ldots, \mathcal{O}_{P'_n}}$ which simulates $\mathcal{A}^{\mathcal{O}, \mathcal{O}_{C[P_1(\overline{t_1}),\ldots,P_n(\overline{t_n})]?C[P_1(\overline{t'_1}),\ldots,P_n(\overline{t'_n})]}}$ but, after setting some variables $d_1, \ldots, d_n$ to 0 and some list $\overline{x}$ to the empty list, for every call to $\mathcal{O}_{C[P_1(\overline{t_1}),\ldots,P_n(\overline{t_n})]?C[P_1(\overline{t'_1}),\ldots,P_n(\overline{t'_n})]}$ of the form (c, m):

- if there exist i such that $d_i = 1$ and $c \in \mathcal{C}(P'_i)$ then
  - query $\mathcal{O}_{P'_i}$ with $(c\sigma^{-1}, m)$
  - if $\mathcal{O}_{P'_i}$ terminates set $c_i = 0$ and if it returns $\perp$, then, with $\overline{C}$ and C'' such that $C[\__1, \ldots, \__n] = \overline{C}[\__i; C'']$ it adds to $\overline{x}$ the channels $\mathcal{C}(C'')$
  - else it forwards the answer (c', m') as $(c'\sigma, m')$
- else if $c \in \mathcal{C}(C_1)$ and $c \notin \overline{x}$ then
  - queries $\mathcal{O}_D$ with (c, m)
  - if $\mathcal{O}_D$ answers with some $\overline{t_i}$ on channel i

* set $d_i = 1$
* sends $(i, \overline{t_i})$ to $\mathcal{O}_{P'_i}$ and forwards the answer

– else forwards the answer of $\mathcal{O}_D$

With this construction, we do have

$$\mathrm{Adv}\left(\mathcal{B}^{\mathcal{O},\mathcal{O}_{C_1?C_2},\mathcal{O}_{P'_1},\ldots,\mathcal{O}_{P'_n}}\right) = \epsilon_0$$

Using Lemma 4, we get a distinguisher $\mathcal{B}'$ such that:

$$\mathrm{Adv}\left(\mathcal{B}'^{\mathcal{O},\mathcal{O}_{C_1?C_2},\mathcal{O}_{P'_1\|\ldots\|P'_n}}\right) = \epsilon_0$$

Now, with the fact that $\nu\overline{n}.P'_1\|\ldots\|P'_n$ is $\mathcal{O}$ simulatable, we have a simulator $\mathcal{A}^{\mathcal{O}}_{P'_1\|\ldots\|P'_n}$ such that thanks to Proposition 7, $\mathcal{B}'[\mathcal{A}^{\mathcal{O}}_{P'_1\|\ldots\|P'_n}]^{\mathcal{O},\mathcal{O}_D}$ behaves exactly as $\mathcal{B}^{\mathcal{O},\mathcal{O}_{P'_1\|\ldots\|P'_n},\mathcal{O}_D}$.

We finally have $\mathrm{Adv}\left(\mathcal{B}'[\mathcal{A}^{\mathcal{O}}_{P'_1\|\ldots\|P'_n}]^{\mathcal{O},\mathcal{O}_{C_1?C_2}}\right) = \epsilon_0$.

The bound on the advantage is derived similarly to Theorem 2.

□

When we do so, we only assume that they are all distinct. The following example shows how Theorems 2 and 4 can be used to derive the security of one session of a key exchange composed with a protocol.

Example 4.4. Let us consider a key exchange $I\|R$ where x^I (resp. x^R) is the key derived by the initiator I (resp. the responder R) in case of success. We denote by $KE[\__1, \__2] := I; \__1\|R; \__2$ the composition of the key exchange with two continuations; the binding of x^I (resp. x^R) is passed to the protocol in sequence. Consider possible continuations $P^I(x^I), P^R(x^R)$ that use the derived keys and ideal continuations (whatever "ideal" is) $Q^I(x^I), Q^R(x^R)$. We sketch here how to prove $KE[P^I(x^I), P^R(x^R)] \cong KE[Q^I(x^I), Q^R(x^R)]$ (i.e., the security of the channel established by the key exchange). This will be generalized to multi-sessions in Section 6. We use both Theorems 2 and 4.

Assume, with a fresh name k, that:

1. $\mathcal{O}_{ke}$ is an oracle allowing to simulate the key exchange
2. $\mathcal{O}_{P,Q}$ allows to simulate $\mathtt{in}(c_I, x).P^I(x)\|\mathtt{in}(c_R, x).P^R(x)$ and $\mathtt{in}(c_I, x).Q^I(x)\|\mathtt{in}(c_R, x).Q^R(x)$
3. $P^I(k)\|P^R(k) \cong_{\mathcal{O}_{ke}} Q^I(k)\|Q^R(k)$
4. $KE[\mathtt{out}(c_I, x^I), \mathtt{out}(c_R, x^R)] \cong_{\mathcal{O}_{P,Q}} KE[\mathtt{out}(c_I, k), \mathtt{out}(c_R, k)]$

Hypothesis 3 captures the security of the channel when executed with an ideal key, and Hypothesis 4 captures the security of the key exchange. Both indistinguishability are for an attacker that can simulate the other part of the protocol.

Using Theorem 2 with Hypothesis 1 and 3 yields

$$KE[P^I(k), P^R(k)] \cong KE[Q^I(k), Q^R(k)]$$

Hypothesis 2 and 4 yield, with two applications of Theorem 4, one for P and one for Q, that $KE[P^I(x^I), P^R(x^R)] \cong KE[P^I(k), P^R(k)]$ and $KE[Q^I(x^I), Q^R(x^R)] \cong KE[Q^I(k), Q^R(k)]$. Transitivity allows us to conclude that the key exchange followed by the channel using the produced key is indistinguishable from the key exchange followed by the ideal secure channel:

$$KE[P^I(x^I), P^R(x^R)] \cong KE[Q^I(x^I), Q^R(x^R)]$$

In Theorem 4, the simulatability of

$$\nu\overline{n}.\mathtt{in}(c_P, k); P(k) \| \mathtt{in}(c_Q, k); Q(k)$$

may be a requirement too strong in some applications. This issue will be raised when we consider the forwarding agent of the SSH protocol, as detailed in Section 9.3, but we can avoid it in this specific case. For more complex applications, it might be interesting in the future to consider a weaker version of function applications where the produced key k always satisfies a condition $H(k)$. We could then design an oracle $\mathcal{O}$ so that for all names satisfying condition $H(k)$ we would have that $P(k)\|Q(k)$ is $\mathcal{O}$-simulatable.

4.3 Unbounded Replication

An important feature of a compositional framework is the ability to derive the security of a multi session protocol from the analysis of a single session. To refer to multiple sessions of a protocol, we consider that each session uses some fresh randomness that we see as a local session identifier.

The main idea behind the Theorem is that the oracle will depend on a sequence of names of arbitrary length. This sequence of names represents the list of honest randomness sampled by each party of the protocol, and the oracle enables simulatability of those parties.

We provide bellow the Proposition that allows to put in parallel any number of replications of simulatable protocols.

Proposition 17. *Let $\mathcal{O}_r$ be an oracle parameterized by a sequence of names $\overline{s}$, and $\mathcal{O}$ an oracle. Let $\overline{p}$ be a sequence of names, $P(\overline{x})$, $R_i^1(\overline{x}, \overline{y}), \ldots, R_i^k(\overline{x}, \overline{y})$ and $Q(\overline{x})$ be protocols, such that $\mathcal{N}_l(R_i^1, \ldots, R_i^k)$ is disjoint of the oracle support. If we have, for sequences of names $\overline{lsid}^1, \ldots, \overline{lsid}^k$, with $\overline{s} = \{\overline{lsid}_i^j\}_{1 \leq j \leq k, i \in \mathbb{N}}$:*

1. *$\forall i, j \in \mathbb{N}, \nu\overline{p}, \overline{lsid}_i^j.R_i^j(\overline{p}, \overline{lsid}_i^j)$ is $\mathcal{O}_r$-simulatable.*
2. *$P(\overline{p}) \cong_{\mathcal{O}_r} Q(\overline{p})$*
3. *$\overline{s}$ is disjoint of the support of $\mathcal{O}$.*

Then, for any integers $N_1, \ldots, N_k$:

$$\begin{aligned} &P(\overline{p}) \|^{i \leq N_1} (R_i^1(\overline{p}, \overline{lsid}_i^1) \| \ldots \|^{i \leq N_k} R_i^k(\overline{p}, \overline{lsid}_i^k) \\ &\quad \cong_{\mathcal{O}, \mathcal{O}_r} Q(\overline{p}) \|^{i \leq N_1} R_i^1(\overline{p}, \overline{lsid}_i^1) \| \ldots \|^{i \leq N_k} R_i^k(\overline{p}, \overline{lsid}_i^k) \end{aligned}$$

Specifically, there exists a polynomial p_S (independent of all R^j) such that if p_{R^j} is the polynomial bound on the runtime of the simulator for R^j, we have,

$$\begin{aligned} &\mathrm{Adv}^{P(\overline{p}) \|^{i \leq N_1} (R_i^1(\overline{p}, \overline{lsid}_i^1) \| \ldots \|^{i \leq N_k} R_i^k(\overline{p}, \overline{lsid}_i^k) \cong_{\mathcal{O}} Q(\overline{p}) \|^{i \leq N_1} R_i^1(\overline{p}, \overline{lsid}_i^1) \| \ldots \|^{i \leq N_k} R_i^k(\overline{p}, \overline{lsid}_i^k)}(t) \\ &\quad \leq \mathrm{Adv}^{P(\overline{p}) \cong_{\mathcal{O}, \mathcal{O}_r} Q(\overline{p})} \Big(p_S\big(t, N_1, |R^1|, \ldots, N_k, |R^k|, p_{R^1}(t), \ldots, p_{R^k}(t)\big) \Big) \end{aligned}$$

In the previous proposition and following applications, we talk about sequences of names of the form $\overline{s} = \{\overline{lsid}_i^j\}_{1\leq j\leq k, i\in\mathbb{N}}$. This does not have any practical meaning and is only a shortcut. In practice, we must have that the previous hypotheses hold for any polynomial p and any sequence $\overline{s} = \{\overline{lsid}_i^j\}_{1\leq j\leq k, 1\leq i\leq p(\eta)}$. We will precisely define this in Section 12.

Applying the previous Proposition with P and Q as R^1 and R^2, we can obtain the Theorem for the unbounded replication of a protocol, where the number of sessions depends on the security parameter.

Theorem 5. *Let $\mathcal{O}_r$, $\mathcal{O}$ be oracles both parameterized by a sequence of names $\overline{s}$. Let $\overline{p}$ be a sequence of names, $P_i(\overline{x},\overline{y})$ and $Q_i(\overline{x},\overline{y})$ be parameterized protocols, such that $\mathcal{N}_l(P,Q)$ is disjoint of the oracles support. If we have, for sequences of names $\overline{lsid}^P, \overline{lsid}^Q$, with $\overline{s} = \{\overline{lsid}_i^P, \overline{lsid}_i^Q\}_{i\in\mathbb{N}}$:*

1. *$\forall\ i \geq 1, \nu\overline{p}, \overline{lsid}_i^P.P_i(\overline{p}, \overline{lsid}_i^P)$ is $\mathcal{O}_r$-simulatable.*
2. *$\forall\ i \geq 1, \nu\overline{p}, \overline{lsid}_i^Q.Q_i(\overline{p}, \overline{lsid}_i^Q)$ is $\mathcal{O}_r$-simulatable.*
3. *$\overline{s}$ is disjoint of the support of $\mathcal{O}$.*
4. *$P_0(\overline{p}, \overline{lsid}_0^P) \cong_{\mathcal{O}_r,\mathcal{O}} Q_0(\overline{p}, \overline{lsid}_0^Q)$*

then,

$$||^i P_i(\overline{p}, \overline{lsid}_i^P) \cong_{\mathcal{O}} ||^i Q_i(\overline{p}, \overline{lsid}_i^Q)$$

To prove this result, we use the explicit advantages that can be derived from our composition Theorems, which increases polynomially with respect to the number of sessions, and apply a classical hybrid argument to conclude.

In our applications (Section 6), the main idea is to first use Theorem 5 to reduce the multi-session security of a key exchange or a communication channel to a single session, and then use Theorems 2 and 4 to combine the multiple key exchanges and the multiple channels.

Remark, that in practice, to express the security properties of the protocols, we need to allow the protocols to use a predicate $T(x)$ whose interpretation may depend on the list of honest randomness sampled by each party of the protocol. For instance, this predicate may be used to check whether a value received by a party corresponds to a randomness sent by another party, and we would have $T(x) := x \in \overline{s}$. The two previous Theorems are in fact also valid in such cases, and we will use such notations in the application to key exchanges, but we delay to Section 12 the formalization of such predicates.

5 Unbounded Sequential Replication

We replicate a sequential composition where at each occurrence, a value produced by the protocol is transmitted to the next occurrence. This corresponds to the security of a protocol looping on itself, as it is the case for some key renewal protocols.

Such protocols depend on an original key, and are thus parameterized process of the form $P(x)$. As they renew the key stored in the variable x, they rebind x to some new value and thus contain a construct of the form $\mathsf{let}\ x = \_\ \mathsf{in}\ .$

Proposition 18. *Let $\mathcal{O}$ be an oracle, two parameterized processes $P(x), Q(x)$, a set of names $\overline{n} = \mathcal{N}_g(P,Q)$ and fresh names k_0, l. We assume that $\mathcal{N}_l(P,Q)$ is disjoint of the support of $\mathcal{O}$. If:*

- $\nu\overline{n}.\mathtt{in}(c_P, x); P(x) \| \mathtt{in}(c_Q, x); Q(x)$ *is $\mathcal{O}$-simulatable, and*
- $P(k_0); \mathtt{out}(c_P, x) \| Q(k_0); \mathtt{out}(c_Q, x) \cong_{\mathcal{O}} P(k_0); \mathtt{out}(c_P, l) \| Q(k_0); \mathtt{out}(c_Q, l)$

then, for any N,

$$\begin{aligned} &P(k_0); P(x)^{;N}; \mathtt{out}(c_P, x) \| Q(k_0); Q(x)^{;N}; \mathtt{out}(c_Q, x) \\ &\quad \cong_{\mathcal{O}} P(k_0); P(x)^{;N}; \mathtt{out}(c_P, l) \| Q(k_0); Q(x)^{;N}; \mathtt{out}(c_Q, l) \end{aligned}$$

The main idea behind the proof is to perform as many function applications (Theorem 4) as needed, one for each replication of the protocol. Remark that compared to the previous replication, where we considered multiple sessions of the protocol and thus a notion of local session identifier was required, here we consider a single session looping on itself, and we do not need those identifiers.

Part II
Applications to Key Exchange

6 Application to Key Exchanges

Although our framework is not specifically tailored to key exchanges or any specific property, we choose to focus here on this application. We outline how our theorems may be used to prove the security of a protocol using a key derived by a key exchange in a compositional way. (Let us recall that the key exchange and the protocol using the derived key may share long term secrets).

6.1 Our Model of Key Exchange

In order to obtain injective agreement, key exchanges usually use fresh randomness for each session as local session identifiers. For instance in the case of a Diffie-Hellman key exchange, the group shares may be seen as local session identifiers.

As in Example 4.4, KE is a key exchange with possible continuations. In addition, we consider multiple copies of KE, indexed by i, and local session identifiers $lsid$ for each copy:

$$KE_i[\__1, \__2] := I(lsid_i^I, id^I); \__1 \| R(lsid_i^R, id^R); \__2$$

Here, id captures the identities of the parties and $lsid$ captures the randomness that will be used by I and R to derive their respective local session identifiers. In the key exchange, I binds x^I to the key that it computes, x^I_{lsid} to the value of $lsid$ received from the other party and x^I_{id} to the received identity. Symmetrically, R binds the variables x^R, x^R_{lsid} and x^R_{id}.

If we denote by $P_i^I(x^I) \| P_i^R(x^R)$ the continuation (e.g., a record protocol based on the derived secret key), $KE_i[P_i^I(x^I), P_i^R(x^R)]$ is the composition of a session of the key exchange with the protocol where the values of x^I, x^R (computed keys) are passed respectively to

$P_i^I(x^I)$ or $P_i^R(x^R)$. With Q an idealized version of P (however it is defined), the security of the composed protocol is expressed as follows:

$$\|^i KE_i[P_i^I(x^I), P_i^R(x^R)] \cong \|^i KE_i[Q_i^I(x^I), Q_i^R(x^R)]$$

Intuitively, from the adversary point of view, P is equivalent to its idealized version, even if the key is derived from the key exchange as opposed to magically shared.

Equivalently, the security of the composed protocol can be proved if we have that the advantage against the following indistinguishability is polynomial in N (and of course negligible).

$$\|^{i\leq N} KE_i[P_i^I(x^I), P_i^R(x^R)] \cong \|^{i\leq N} KE_i[Q_i^I(x^I), Q_i^R(x^R)]$$

A Corollary formalizing the following discussion can be found in Appendix F.1.

6.2 Proofs of Composed Key Exchange Security

Following the same applications of Theorems 2 and 4 as in Example 4.4, we decompose the proof of the previous indistinguishability goals into the following goals:

1. find an oracle $\mathcal{O}_{P,Q}$ to simulate multiple sessions of P or Q,
2. design an oracle $\mathcal{O}_{ke}$ to simulate multiple sessions of KE
3. complete a security proof under $\mathcal{O}_{ke}$ for multiple sessions of the protocol using fresh keys,
4. complete a security proof under $\mathcal{O}_{P,Q}$ for multiple sessions of the key exchange.

We further reduce the security of the protocol to smaller proofs of single sessions of the various components of the protocols under well chosen oracles. The following paragraphs successively investigate how to simplify the goals (1),(2),(3),(4) above. For simplicity, we only consider here the case of two fixed honest identities.

In the following, we provide the conditions S-1,S-2,P-1,P-2,P-3,P-4,K-1,K-2,K-3 that must be satisfied, so that we can prove

$$\|^i KE_i[P_i^I(x^I), P_i^R(x^R)] \cong \|^i KE_i[Q_i^I(x^I), Q_i^R(x^R)]$$

using our framework and the decomposition of Example 4.4. Corollary 2, that formalizes the following discussion and generalizes it to non fixed identities, can be found in Appendix F.1.

We denote $\overline{p} = \{id^I, id^R\}$ and assume that they are the only shared names between KE, P and Q and are the only names shared by two distinct copies P_i, P_j (resp. Q_i, Q_j). We also denote by $\overline{s} = \{lsid_i^I, lsid_i^R\}_{i\in\mathbb{N}}$ the set of all copies of the local session identifiers.

Protocol simulatability For the simulation of the protocol, there must exists an oracle $\mathcal{O}_{P,Q}$ such that

$$\text{S-1 } \nu\overline{p}.\mathtt{in}(c_I, x^I).P_i^I(x^I)\|\mathtt{in}(c_R, x^R).P_i^R(x^R) \text{ is } \mathcal{O}_{P,Q}\text{-simulatable}$$

Indeed, if this condition is fulfilled (and a similar one replacing P with Q), then, thanks to Theorem 1, $\nu\overline{p}.\|^i(\mathtt{in}(c_I, x^I).P_i^I(x^I)\|\mathtt{in}(c_R, x^R).P_i^R(x^R))$ is $\mathcal{O}_{P,Q}$-simulatable (and similarly for Q). This meets the condition (2) of Theorem 4.

Key exchange simulatability For the simulation of the key exchange context, we need N (with N polynomial in the security parameter) copies of KE and, in each of them, the initiator (resp. the responder) may communicate with N possible responders (resp. initiators). We therefore use Theorem 2 with a context C with $2N^2$ holes. C is the parallel composition of N contexts and, as above, we use Theorem 1 to get the condition (1) of Theorem 2. Let KE'_i be[4]

$$KE_i[\ \mathsf{if}\ \bigvee_{1\le j\le N} x^I_{lsid} = lsid^R_j\ \mathsf{then}\ \mathtt{out}(c_I, \langle i,j\rangle)\ \mathsf{else}\ \bot,$$
$$\mathsf{if}\ \bigvee_{1\le j\le N} x^R_{lsid} = lsid^I_j\ \mathsf{then}\ \mathtt{out}(c_R, \langle i,j\rangle)\ \mathsf{else}\ \bot]$$

$\overline{C}$ is then $\|^{i\le N} KE'_i$ and C can be inferred by replacing each $\mathtt{out}(\langle i,j\rangle)$ with a hole. We output $\langle i,j\rangle$ so that we know that the full scheduling is simulatable. Then, the condition to be met by the key exchange is that

S-2 $\nu\overline{p}.KE'_i$ is $\mathcal{O}_{ke}$-simulatable

We then get, thanks to Theorem 1 the condition (1) of Theorem 2.

Security of the protocol Our goal is $\|^i P_i(k_i) \cong_{\mathcal{O}_{ke}} \|^i Q_i(k_i)$. Based on Theorem 5, we only need an oracle $\mathcal{O}_r$ so that:

P-1) $\forall\ i \ge 1, \nu\overline{p}, k_i.P_0(k_i)$ is $\mathcal{O}_r$-simulatable,

P-2) $\forall\ i \ge 1, \nu\overline{p}, k_i.Q_0(k_i)$ is $\mathcal{O}_r$-simulatable,

P-3) $\overline{s}$ is disjoint of the support of $\mathcal{O}_{ke}$,

P-4) $P_0(k_0) \cong_{\mathcal{O}_r,\mathcal{O}_{ke}} Q_0(k_0)$.

We use the fresh names k_i to model fresh magically shared keys, and use them as local sids for Theorem 5. The intuition is similar to the notion of Single session game of [18], where the considered protocols are such that we can derive the security of multiple sessions from one session. For instance, if the key is used to establish a secure channel, revealing the other keys does not break the security of one session, but allows to simulate the other sessions.

Security of the key exchange The security of the key exchange is more complicated to define, in the sense that it cannot simply be written with a classical replication. The partnering of sessions is not performed beforehand, so we must consider all possibilities. We may express the security of a key exchange by testing the real-or-random for each possible session key. We denote $k_{i,j}$ the fresh name corresponding to the ideal key that will be produced by the i-th copy of the initiator believing to be partnered with the j-th copy of the responder. The security of the key exchange is captured through the following indistinguishability:

$$\|^{i\le N} KE_i[\mathtt{out}(x^I), \mathtt{out}(x^R)] \cong_{\mathcal{O}_{P,Q}} \|^{i\le N}\ KE_i[\ \mathsf{if}\ \bigvee_{1\le j\le N} x^I_{lsid} = lsid^R_j\ \mathsf{then}\ \mathtt{out}(k_{i,j})\ \mathsf{else}\ \bot,$$
$$\mathsf{if}\ \bigvee_{1\le j\le N} (x^R_{lsid} = lsid^I_j)\ \mathsf{then}\ \mathtt{out}(k_{j,i})\ \mathsf{else}\ \bot]$$

where the advantage of the attacker is polynomial in N. Remark that we sometimes omit channels, when they only need to be distinct.

[4] we denote $\mathsf{if}\ \bigvee_{1\le j\le N} c_i\ \mathsf{then}\ a_i\ \mathsf{else}\ a' := \mathsf{if}\ c_1\ \mathsf{then}\ a_1\ \mathsf{else if}\ c_2 \cdots\ \mathsf{then}\ a_n\ \mathsf{else}\ a'$

Using a classical cryptographic hybrid argument (detailed in Proposition 39), we reduce the security of multiple sessions to the security of one session in parallel of multiple corrupted sessions; the security of each step of the hybrid game is derived from Equation (1) using Theorem 4. It is expressed, with $state_i^X = \langle x^X, lsid_i^X, x_{lsid}^X \rangle$, as

$$\begin{aligned}\|^{i\leq N} KE_i[\mathtt{out}(\langle state_i^I\rangle), \mathtt{out}(\langle state_i^R\rangle)] \cong_{\mathcal{O}_{P,Q}} \\ \|^{i\leq N-1} KE_i[\mathtt{out}(\langle state_i^I\rangle), \mathtt{out}(\langle state_i^R\rangle)] \\ \| \ KE_N[\ \mathsf{if}\ x_{lsid}^I = lsid_N^R \ \mathsf{then}\ \mathtt{out}(\langle k, lsid_N^I, x_{lsid}^I\rangle) \\ \mathsf{else\ if}\ x_{lsid}^I \notin \{lsid_i^R\}_{1\leq i\leq N-1}\ \mathsf{then}\ \bot, \\ \mathsf{else}\ \mathtt{out}(\langle state_i^I\rangle), \\ \mathsf{if}\ x_{lsid}^R = lsid_N^I\ \mathsf{then}\ \mathtt{out}(\langle k, lsid_N^R, x_{lsid}^R\rangle) \\ \mathsf{else\ if}\ x_{lsid}^R \notin \{lsid_i^I\}_{1\leq i\leq N-1}\ \mathsf{then}\ \bot, \\ \mathsf{else}\ \mathtt{out}(\langle state_i^R\rangle)]\end{aligned} \tag{1}$$

The previous equivalence expresses that when we look at N sessions that all output their full state upon completion, the particular matching of the parties in KE_N has a key that is real or random if they are indeed partnered together, and if they are not partnered together, they must be talking to another agent from the other KE_i. We may see the other sessions as corrupted sessions, as they leak their states upon completion.

We further reduce the problem to proving the security of a single session even when there is an oracle simulating corrupted sessions. To this end, we need to reveal the dishonest local session's identifiers to the attacker, but also to allow him to perform the required cryptographic operations, e.g. signatures using the identities.

We define, for $X \in \{I, R\}$, $\overline{s}^X$ as the set of copies of the local session identifiers of I or R, except a distinguished one (indexed 0 below) and $\overline{s} = \overline{s}^I \cup \overline{s}^R$. To obtain the security of multiple sessions of the key exchange, we use Proposition 17.[5]. To this end, we would need to design an oracle $\mathcal{O}_r$, such that the following assumptions are satisfied, where $\mathcal{O}_{P,Q}$ corresponds to $\mathcal{O}$ of Proposition 17:

K-1) $\forall 1 \leq i \leq N, \nu lsid_i^I, id^I, lsid_i^R, id^R.$
$KE_i[\mathtt{out}(x^I), \mathtt{out}(x^R)] \| \mathtt{out}(\langle lsid_i^R, lsid_i^I\rangle)$ is $\mathcal{O}_r$ simulatable.

K-2) $KE_0[\ \mathtt{out}(\langle x^I, lsid_0^I, x_{lsid}^I\rangle), \ \mathtt{out}(\langle x^R, lsid_0^R, x_{lsid}^R\rangle)] \ \cong_{\mathcal{O}_r, \mathcal{O}_{P,Q}} \ KE_0[$
$\mathsf{if}\ x_{lsid}^I = lsid_0^R\ \mathsf{then}\ \mathtt{out}(\langle k, lsid_0^I, x_{lsid}^I\rangle)$
$\mathsf{else\ if}\ x_{lsid}^I \notin \overline{s}^R\ \mathsf{then}\ \bot$
$\mathsf{else}\ \mathtt{out}(\langle x^I, lsid_0^I, x_{lsid}^I\rangle),$
$\mathsf{if}\ x_{lsid}^R = lsid_0^I\ \mathsf{then}\ \mathtt{out}(\langle k, lsid_0^R, x_{lsid}^R\rangle)$
$\mathsf{else\ if}\ x_{lsid}^R \notin \overline{s}^I\ \mathsf{then}\ \bot$
$\mathsf{else}\ \mathtt{out}(\langle x^R, lsid_0^R, x_{lsid}^R\rangle)]$

K-3) $\overline{s}$ is disjoint of the support of $\mathcal{O}_{P,Q}$.

Intuitively, if the initiator believes to be talking to the honest responder, then it outputs the ideal key, and if it is not talking to any simulated corrupted party, it raises a bad event.

Note that while the structure of the proof does not fundamentally change from other proofs of key exchanges, e.g. [18], each step of the proof becomes straightforward thanks to our composition results. Our proofs are also more flexible, as shown by the extension to key exchanges with key confirmation in Section 8.

[5]We also use Theorem 1 to get the simulatability of N sessions in parallel from the simulatability of each session.

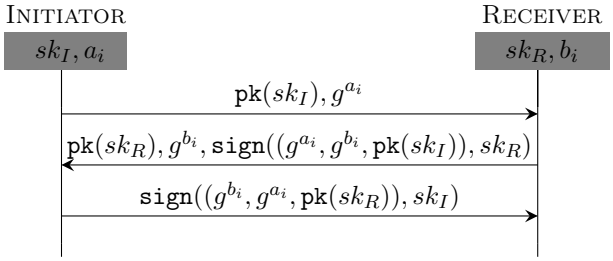

Figure 4: ISO 9798-3 Diffie Hellman Key Exchange

7 Basic Diffie-Hellman Key Exchange

We outline here the application of our framework to the ISO 9798-3 protocol, a variant of the Diffie-Hellman key exchange. It is proven UC composable in [13]. We use our result to extend the security proof to a context with shared long term secrets (which was not the case in the UC proof). We present the protocol in Figure 4, and show how to instantiate the required values and oracles to perform the proof presented in Section 6.2. The formal proofs (using the CCSA model [11]) are provided in Appendix C.

Our decomposition and subsequent proofs show that the DDH key exchange can be used to securely derive a secret key for any protocol that does not rely on the long term secret used in the key exchange. Our proof is also modular, in the sense that it could be adapted to provide also the security when the continuation protocol uses the long term shared secret as well.

A high level view of the protocol is given in Figure 4, and it is formally expressed in our algebra in Figure 5, where $\_I$ and $\_R$ denote the possible continuations at the end of each party. We use pattern matching in the inputs to simplify the notations, where for instance $\mathtt{in}(c, \langle m, x\rangle)$ with m some constant only accepts inputs whose first projection is m, and then bind the variable x to the second projection. If the inputs are not of the given form, the protocols goes to an error branch.

Our goal is to apply the decomposition of Section 6.2, for some abstract continuations P and Q that are supposed to used the derived key. We need to find suitable identities and local session identifiers so that the Conditions from the decomposition of Section 6.2 are fulfilled. As we do not specify P and Q, we only discuss the conditions relative to the security of the key exchange, e.g., K-1,K-2 and K-3. Remark that those conditions are sufficient to derive a notion similar to the classical security of a key exchange, as for any P and Q that do not share long term shared secrets with the key exchange. The other conditions are trivial to derive or only rely on the security of the continuation when using an ideal key.

The identity of each party is its long term secret key, and thus, we use sk_I and sk_R as id_I and id_R. Each session of the key exchange instantiates a fresh Diffie-Hellman share, that can be seen as a local session identifier. We thus use g^{a_i} and g^{b_i} as $lsid_i^I$ and $lsid_i^R$. These values can also be used as implicit tagging since any signed message either depends on a_i or b_i.

With those choices, we need to find a tagging function T that will provide a tagged oracle $\mathcal{O}_T$ such that the Conditions K of Section 6.2 are satisfied. Those Conditions, reformulated with the current notations and with $\mathcal{O}_T$ standing for $\mathcal{O}_r$, are expressed as follow:

K-1) $\forall 1 \leq i \leq N, \nu a_i, sk_I, b_i, sk_R.$
$I_i[\mathtt{out}(k_I)] \| R_i[\mathtt{out}(k_R)] \| \mathtt{out}(\langle g^{a_i}, g^{b_i}\rangle)$ is $\mathcal{O}_T$-simulatable.

$$
\begin{array}{l}
\|^i \quad (\\
\qquad I_i := \\
\qquad\qquad \mathtt{out}(\langle \mathtt{pk}(sk_I), g^{a_i} \rangle) \\
\qquad\qquad \mathtt{in}(\langle x_{pk}, x_B, x_m \rangle). \\
\qquad\qquad \mathsf{if}\ \mathtt{verify}(x_m, x_{pk}) = \langle g^{a_i}, x_B, \mathtt{pk}(sk_I) \rangle\ \mathsf{then} \\
\qquad\qquad\qquad \mathtt{out}(\mathtt{sign}(\langle x_B, g^{a_i}, x_{pk} \rangle, sk_I)) \\
\qquad\qquad\qquad \mathsf{let}\ k_I = x_B^{a_i}\ \mathsf{in} \\
\qquad\qquad\qquad -_I \\
\qquad \| \\
\qquad R_i := \\
\qquad\qquad \mathtt{in}(\langle x_{pk}, x_A \rangle). \\
\qquad\qquad \mathtt{out}(\langle \mathtt{pk}(sk_R), g^{b_i}, \mathtt{sign}(\langle x_A, g^{b_i}, x_{pk} \rangle, sk_R) \rangle) \\
\qquad\qquad \mathtt{in}(x_m). \\
\qquad\qquad \mathsf{if}\ \mathtt{verify}(x_m, x_{pk}) = \langle g^{b_i}, x_A, \mathtt{pk}(sk_R) \rangle\ \mathsf{then} \\
\qquad\qquad\qquad \mathsf{let}\ k_R = x_A^{b_i}\ \mathsf{in} \\
\qquad\qquad\qquad -_R \\
\quad)
\end{array}
$$

Figure 5: ISO 9798-3 Diffie Hellman Key Exchange in the Pi Calculus (omitted channels)

$$
\text{K-2)} \quad \begin{array}{r} I_0\ [\mathtt{out}(\langle k_I, g^{a_0}, x_B \rangle)] \\ \|R_0\ [\mathtt{out}(\langle k_R, g^{b_0}, x_A \rangle)] \end{array} \cong_{\mathcal{O}_T, \mathcal{O}_{P,Q}} \begin{array}{r} I_0 \left[\begin{array}{l} \mathsf{if}\ x_B = g^{b_0}\ \mathsf{then}\ \mathtt{out}(\langle x_B^{a_0}, g^{a_0}, x_B \rangle) \\ \mathsf{else\ if}\ x_B \notin \{g^{b_i}\}_{i \geq 1}\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathtt{out}(\langle k_I, g^{a_0}, x_B \rangle) \end{array} \right] \\ \|R_0 \left[\begin{array}{l} \mathsf{if}\ x_A = g^{a_0}\ \mathsf{then}\ \mathtt{out}(\langle x_A^{b_0}, g^{b_0}, x_A \rangle) \\ \mathsf{else\ if}\ x_A \notin \{g^{a_i}\}_{i \geq 1}\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathtt{out}(\langle k_R, g^{a_0}, x_B \rangle) \end{array} \right] \end{array}
$$

K-3) $\{g^{a_i}, g^{b_i}\}_{i \geq 1}$ is disjoint of the support of $\mathcal{O}_{P,Q}$.

K-2 either corresponds to a matching conversation (i.e., all messages received by one were sent by the other) between the sessions with sids g^{a_0}, g^{b_0}, in which case the output is (twice) an ideal key k, or else it is a matching conversation with a simulated session, in which case it outputs the computed keys. It is neither of those cases, it should not happen, and we raise a bad event (denoted $\bot$). The proof of the K-2 is thus a real-or-random proof of a honestly produced key. We do not provide the proof of K-2 in this section, it is provided in Appendix C.

We must define an implicit tagging that allows to both have the simulatability and the indistinguishability. Remark that first, we extend the tagging function T of Definition 14 so that it may depend on a second argument of arbitrary length, yielding $T(m, \overline{s})$, the corresponding signing oracle being denoted $\mathcal{O}^{\mathtt{sign}}_{T,sk,\overline{s}}$. This is required so that the implicit tagging may depend on all the possible local session identifiers. The exact definition of this extension is given in Section 12.

We define the implicit tagging functions T^I and T^R as

$$
\begin{array}{l}
T^I(m, \{g^{a_i}, g^{b_i}\}_{i \geq 1}) := \ \exists s \in \{a_i\}_{i \geq 1}, \exists m_1, m_2 . m = (m_1, g^s, m_2) \\
T^R(m, \{g^{a_i}, g^{b_i}\}_{i \geq 1}) := \ \exists s \in \{b_i\}_{i \geq 1}, \exists m_1, m_2 . m = (m_1, g^s, m_2)
\end{array}
$$

This tagging function will suit our needs, as all messages signed by the two parties follow this pattern. Moreover, in the protocol, the value sent in the first message should match g^{a_i} in

the last message. Therefore, when the protocol of Figure 4 is successfully completed, we can prove that if $x_B \neq g^{b_0}$, then $x_B \in \{g^{b_i} | i \geq 1\}$, i.e., $T^R(x_B, \{g^{a_i}, g^{b_i}\}_{i\geq 1})$ is true (and similarly for R).

Let $\overline{s} = \{g^{a_i}, g^{b_i}\}_{i\geq 1}$, we finally define $\mathcal{O}_T = \mathcal{O}^{\mathtt{sign}}_{T^I, sk_I, \overline{s}}, \mathcal{O}^{\mathtt{sign}}_{T^R, sk_R, \overline{s}}, \mathcal{O}_{\overline{s}}$, where $\mathcal{O}_{\overline{s}}$ simply reveals the elements in $\overline{s}$, we do obtain the simulatability of multiple sessions of the key exchange (Hypothesis 1).

To adapt this proof to a concrete example, the security proof of K-2 would be performed under an oracle $\mathcal{O}_{P,Q}$ that allows to simulate the continuation (Condition P-1 of Section 6.2). The continuation should then be proven secure when using an ideal key (Conditions P of Section 6.2). In some cases, this step is trivial. Indeed, let us consider a record protocol $L := L^I(x^I) \| L^R(x^R)$, that exchanges encrypted messages using the exchanged key, and does not share any long term secret, i.e., does not use the signing keys of the key exchange. Without any shared secret, we do not need any oracle to simulate $\mathtt{in}(k); L^I(k) \| \mathtt{in}(k); L^R(k)$, so we can choose a trivial $\mathcal{O}_{P,Q}$ that does nothing.

8 Extension to Key Confirmations

We present how our compositional framework can be used to prove the security of a key exchange, in which the key is derived in a first part of the protocol and then used (key confirmation) in the second part. Compared to [8], our method allows in addition sharing of long term secrets.

Consider a key exchange $I(lsid_i^I, id^I) \| R(lsid_i^R, id^R)$. We further split I and R into $I_i := I_i^0(lsid_i^I, id^I); I_i^1(x^I)$ and $R_i := R_i^0(lsid_i^R, id^r); R_i^1(x^R)$, where I_i^0 and R_i^0 correspond to the key exchange up to, but not including, the first use of the secret key (x^I or x^R), and I_i^1 and R_i^1 are the remaining parts of the protocol. The intuition behind the proof of security is that at the end of I_i^0 and R_i^0, i.e. just before the key confirmation, either the sessions are partnered together and the derived key satisfies the real-or-random, or they are not, which means that the key confirmation performed by I_i^1 and R_i^1 will fail. We denote

$$KE_i[\__1, \__2] := I_i^0(lsid_i^I, id^I); I_i^1(x^I); \__1 \| R_i^0(lsid_i^R, id^R); R_i^1(x^R); \__2$$

and

$$KE_i^0[\__1, \__2] := I_i^0(lsid_i^I, id^I); \__1 | R_i^0(lsid_j^R, id^R); \__2$$

We proceed as in Section 6, outlining how we may split the security proof into smaller proofs using our framework, using the same composition Theorems at each step. We thus provide the necessary Conditions S-1,S-2,P-1,K-1,K-2,K-3 so that, for some continuation $P_i^I(x^I) \| P_i^R(x^R)$ and its idealized version Q,

$$\|^i KE_i[P_i^I(x^I), P_i^R(x^R)] \cong \|^i KE_i[Q_i^I(x^I), Q_i^R(x^R)]$$

A formal Corollary can be found in Appendix F.2.

8.1 Proofs with Key Confirmations

Key exchange and protocol simulatability We modify slightly the conditions S-1 and S-2 of Section 6.2 to reflect the fact that we now consider the key confirmation to be part of the continuation:

S-1) $\nu\overline{p}.\mathtt{in}(x).I^1(x); P^I(x)$, $\mathtt{in}(x).R^1(x); P^R(x)$, $\mathtt{in}(x).I^1(x); Q^I(x)$, $\mathtt{in}(x).R^1(x); Q^R(x)$ are $\mathcal{O}_{P,Q}$ simulatable.

S-2) $\nu\overline{p}.\ \|^{i\leq N}\ I_i^0(lsid_i^I, id^I);\ \ \mathsf{if}\underset{1\leq i\leq N}{} x_{lsid}^I = lsid_j^R\ \mathsf{then}\ \mathtt{out}(\langle i,j\rangle)\ \mathsf{else}\ I_i^1(x^I); \bot$
$\|^{i\leq N}\ R_i^0(lsid_i^R, id^R);\ \mathsf{if}\underset{1\leq i\leq N}{} x_{lsid}^R = lsid_j^I\ \mathsf{then}\ \mathtt{out}(\langle i,j\rangle)\ \mathsf{else}\ R_i^1(x^R); \bot$
is $\mathcal{O}_{ke}$-simulatable.

Security of the protocol Compared to Section 6.2, the continuation must be secure even in the presence of the messages produced during the key confirmation:

P-1) $\|^{i\leq N} I_i^1(x^I); P_i^I(x^I) \| R_i^1(x^R); P_i^R(x^R) \cong_{\mathcal{O}_r,\mathcal{O}_k} \|^{i\leq N} I_i^1(x^I); Q_i^I(x^I) \| R_i^1(x^R); Q_i^R(x^R)$

We could once again split this goal into a single session proof using Theorem 5. We remark that to prove the security of the single session, we can further reduce the proof by using an oracle that may simulate I^1 and R^1, as the security of P should not depend on the messages of the key confirmation.

Security of the key exchange We proceed in a similar way as in Section 6.2 and we use the same notations. The following Conditions are then suitable:

K-1) $\forall i \leq N, \nu lsid_i^I, id^I, lsid_i^R, id^R.$
$KE_i^0[\mathtt{out}(x^I), \mathtt{out}(x^R)] \| \mathtt{out}(\langle lsid_i^R, lsid_i^I\rangle)$ is $\mathcal{O}_T$-simulatable

K-2) $\overline{s}$ is disjoint of the support of $\mathcal{O}_{P,Q}$.

K-3) $KE_0^0[\mathsf{if}\ x_{lsid}^I \notin \overline{s}^R\ \mathsf{then}\ I_0^1(x^I)\ \mathsf{else}\ \mathtt{out}(\langle x^I, lsid_0^I, x_{lsid}^I\rangle),\ \mathsf{if}\ x_{lsid}^R \notin \overline{s}^I\ \mathsf{then}\ R_0^1(x^R)\ \mathsf{else}\ \mathtt{out}(\langle x^R, lsid_0^R, x_{lsid}^R\rangle)]$
$\cong_{\mathcal{O}_{KE},\mathcal{O}_{P,Q}}$
$KE_0^0[\mathsf{if}\ x_{lsid}^I = lsid_0^R\ \mathsf{then}\ \mathtt{out}(\langle k, lsid_0^I, x_{lsid}^I\rangle)\ \mathsf{else\ if}\ x_{lsid}^I \notin \overline{s}^R\ \mathsf{then}\ I_0^1(x^I); \mathtt{out}(\bot)\ \mathsf{else}\ \mathtt{out}(\langle x^I, lsid_0^I, x_{lsid}^I\rangle),\ \mathsf{if}\ x_{lsid}^R = lsid_0^I\ \mathsf{then}\ \mathtt{out}(\langle k, lsid_0^R, x_{lsid}^R\rangle)\ \mathsf{else\ if}\ x_{lsid}^R \notin \overline{s}^I\ \mathsf{then}\ R_0^1(x^R); \mathtt{out}(\bot)\ \mathsf{else}\ \mathtt{out}(\langle x^R, lsid_0^R, x_{lsid}^R\rangle)]$

The indistinguishability expresses that, if the two singled out parties are partnered, i.e., $x_{lsid}^I = lsid_0^R$ or $x_{lsid}^R = lsid_0^I$, then we test the real-or-random of the key. Else, it specifies that a party must always be partnered with some honest session, i.e., that $x_{lsid}^X \notin \overline{s}^Y$ will never occur. To this end, on one side, when $x_{lsid}^X \notin \overline{s}^Y$ we run the key confirmation, and on the other side we run the key confirmation followed in case of success by a bad event. Finally, when two honest parties are partnered, but are not the singled out parties, they leak their states.

9 Application to SSH

SSH [12] is a protocol that allows users to login onto a server from a remote platform. It is widely used in the version where signatures are used for authentication. An interesting feature

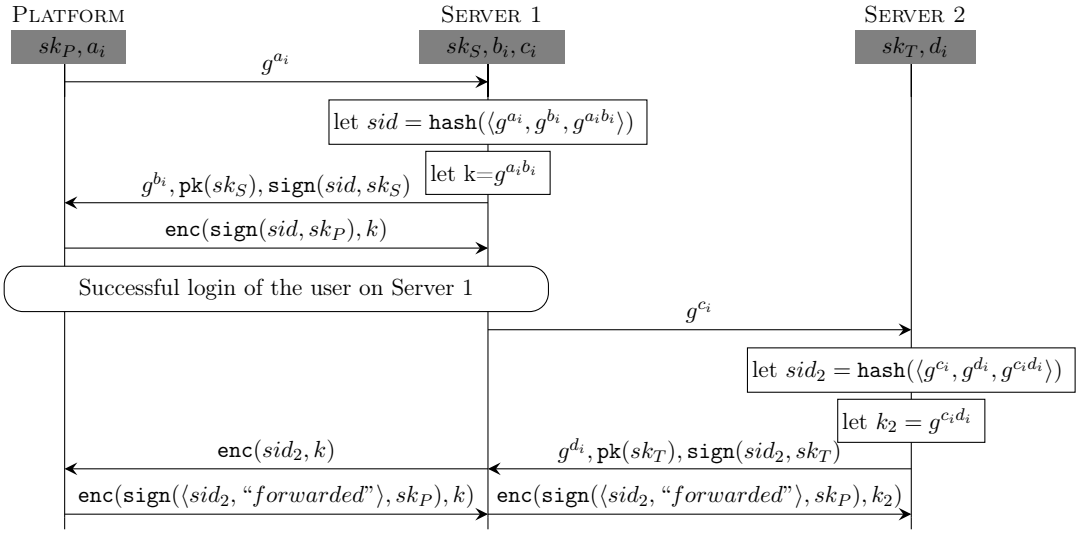

Figure 6: SSH with Forwarding Agent

is forwarding agent: once a user u is logged on a server S, they may, from S, perform another login on another server T. As S does not have access to the signing key of u, it forwards a signature request to u's platform using the secure SSH channel between u and S. This represents a challenge for compositional proofs: we compose a first key exchange with another one, the second one using a signature key already used in the first.

We provide the decomposition of the security proof of SSH composed with one (modified) forwarding agent. We use multiple times in sequence our composition Theorems, that allow us to further simplify the required indistinguishability proofs. The corresponding indistinguishability proofs are performed in Appendix D and Appendix E.

There is a known weakness in this protocol: any privileged user on S can use the agents of any other user as a signing oracle. Thus, in order to be able to prove the security of the protocol, we only consider the case where there is no such privileged user. Figure 6 presents an example of a login followed by a login using the forwarding agent. For simplicity, we abstract away some messages that are not relevant to the security of the protocol.

In the current specification of the forwarding agent, it is impossible for a server to know if the received signature was completed locally by the user's platform, or remotely through the forwarding agent. As the two behaviors are different in term of trust assumptions, we claim that they should be distinguishable by a server. For instance, a server should be able to reject signatures performed by a forwarded agent, because intermediate servers are not trusted. To this end, we assume that the signatures performed by the agent are (possibly implicitly) tagged in a way that distinguishes between their use in different parts of the protocol. This assumption also allows for domain separation between the two key exchanges, and thus simplifies the proof.

We consider a scenario in which there is an unbounded number of sessions of SSH, each with one (modified) forwarding agent, used to provide a secure channel for a protocol P. Thanks to multiple applications of Theorems 2 and 4, we are able to break the proof of this SSH scenario into small ones, that are very close to the proof of a simple Diffie-Hellman key exchange. This assumes the *decisional Diffie-Hellman* (DDH) hypothesis for the group, EUF-CMA for the

$P_i :=$
$\quad \mathtt{out}(g^{a_i});$
$\quad \mathtt{in}(\langle x_B, \mathtt{pk}(sk_S), sign\rangle)$
$\quad \mathsf{let}\ k = x_B^{a_i}\ \mathsf{in}$
$\quad \mathsf{let}\ sid = \mathtt{hash}(\langle g^{a_i}, x_B, k\rangle)\ \mathsf{in}$
$\quad \mathsf{if}\ \mathtt{verify}(sign, \mathtt{pk}(sk_S)) = sid\ \mathsf{then}$
$\quad\quad \mathtt{out}(\mathtt{enc}(\mathtt{sign}(sid, skP), k));$
$\quad\quad \_P.$

$S_i :=$
$\quad \mathtt{in}(x_A);$
$\quad \mathsf{let}\ k = x_A^{b_i}\ \mathsf{in}$
$\quad \mathsf{let}\ sid = \mathtt{hash}(\langle x_A, g^{b_i}, k\rangle)\ \mathsf{in}$
$\quad \mathtt{out}(\langle g^{b_i}, \mathtt{pk}(sk_S), \mathtt{sign}(sid, sk_S)\rangle)$
$\quad \mathtt{in}(\mathtt{enc}(x_{sign}, k))$
$\quad \mathsf{if}\ \mathtt{verify}(x_{sign}, \mathtt{pk}(skP)) = sid\ \mathsf{then}$
$\quad\quad \_S.$

$$SSH := \|^i(P_i[0]\|S_i[0])$$

Figure 7: Basic SSH Key Exchange

signature scheme and that the encryption must ensure integrity of the cyphertexts (this last assumption is only required for the forwarded key exchange, where a signature is performed over an encrypted channel). P also has to satisfy the conditions of Section 8.1. In particular, it must be secure w.r.t. an attacker that has access to a hash that includes the exchanged secret key, since SSH produces such a hash. Note that the scenario includes multiple sessions, but only one forwarding. The extension would require an induction to prove in our framework the security for any number of chained forwardings.

9.1 The SSH Protocol

The basic SSH key exchange is presented in Figure 7, with possible continuations at the end denoted by $\_P$ and $\_S$. In this Section, we use a strong notion of pattern matching, where for instance $\mathtt{in}(\mathtt{enc}(x_{sign}, k))$ is a syntactic sugar for $\mathtt{in}(x); \mathsf{let}\ x_{sign} = \mathtt{dec}(x, k)\ \mathsf{in}\ \_$.

As it is always the case for key exchanges that contain a key confirmation, the indistinguishability of the derived key is not preserved through the protocol. The difficulty of SSH is moreover that once a user has established a secure connection to a server, they can from this server establish a secure connection to another server, while using the secure channel previously established to obtain the user credentials. We provide in Figure 8 a model of the SSH with forwarding of agent (reusing the definitions of P and S from Figure 7). After a session of P terminates successfully, a $ForwardAgent$ is started on the computer. It can receive on the secret channel a signing request and perform the signature of it. In parallel, after the completion of a session of S, a distant session of P that runs on the same machine as S can be initiated by $PDistant$. It will request on the previously established secret channel the signature of the corresponding sid. Finally, as the forwarding can be chained multiple time, at the end of a successful $PDistant$, a $ForwardServer$ is set up. It accepts to receive a signing request on the new secret channel of $PDistant$, forwards the request on the old secret channel, gets the signature and finally forwards it.

The forwarding agent implies a difficult composition problem: we sequentially compose a basic SSH exchange with a second one that uses the derived key and the same long term secret keys. Thus, to be able to prove the security of SSH with forwarding agent, we must be able to handle key confirmations and composition with shared long term secrets.

$PDistant_i(oldk) :=$
  $\mathtt{out}(g^{a_i})$;
  $\mathtt{in}(\langle x_B, \mathtt{pk}(sk_S), sign\rangle)$
  let $k = x_B^{a_i}$ in
  let $sid = \mathtt{hash}(\langle g^{a_i}, x_B, kP\rangle)$ in
  if $\mathtt{verify}(sign, \mathtt{pk}(sk_S)) = sid$ then
    $\mathtt{out}(\mathtt{enc}(sid, oldk))$
    $\mathtt{in}(\mathtt{enc}(sign, oldk))$
    $\mathtt{out}(\mathtt{enc}(sign, k))$
    $\_PD$.

$ForwardAgent(k) :=$
  $\mathtt{in}(\mathtt{enc}(sid, k))$
  $\mathtt{out}(\mathtt{enc}(\mathtt{sign}(\langle sid, \text{“}fwd\text{”}\rangle, skP), k))$

$SForward_i :=$
  $\mathtt{in}(x_A)$;
  let $k = x_A^{b_i}$ in
  let $sid = \mathtt{hash}(\langle x_A, g^{b_i}, k\rangle)$ in
  $\mathtt{out}(\langle g^{b_i}, \mathtt{pk}(sk_S), \mathtt{sign}(sid, sk_S)\rangle)$
  $\mathtt{in}(\mathtt{enc}(sign, k))$
  if $\mathtt{verify}(sign, \mathtt{pk}(skP)) = \langle sid, \text{“}fwd\text{”}\rangle$ then
    $\_SF$

$$SSH_{Forward} := \|^i(P_i[ForwardAgent(k)]\|SForward_i\|S_i[PDistant_i(k)])$$

Figure 8: SSH Key Exchange with Forwarding Agent

9.2 Security of SSH

We show how to prove the Conditions of Section 8 to the basic SSH protocol (without forwarding agent). We provide in Figure 9 the decomposition for key exchanges with key confirmation corresponding to the SSH protocol. We directly specify that P and S may only relate to each other by hard-coding the expected public keys in them. This is the classical behaviour of SSH where a user wants to login on a specific server, and the public key of the user was registered previously on the server.

For some abstract continuation $R^P(x)\|R^S(x)$ and its idealized version $Q^P(x)\|Q^S(x)$, our goal would be to prove that

$P_i^0 :=$
  $\mathtt{out}(g^{a_i})$;
  $\mathtt{in}(x_B)$
  let $k = x_B^{a_i}$ in
  0.
$P_i^1(x_B, k) :=$
  $\mathtt{in}(\langle \mathtt{pk}(sk_S), sign\rangle)$
    let $sid = \mathtt{hash}(\langle g^a, x_B, k\rangle)$ in
    if $\mathtt{verify}(sign, \mathtt{pk}(sk_S)) = sid$ then
      $\mathtt{out}(\mathtt{enc}(\mathtt{sign}(sid, sk_P), k))$
      $\_P$.

$S_i^0 :=$
  $\mathtt{in}(x_A)$;
  let $k = x_A^{b_i}$ in
  let $sid = \mathtt{hash}(\langle x_A, g^{b_i}, k\rangle)$ in
    $\mathtt{out}(g^{b_i})$
$S_i^1(sid, k) :=$
    $\mathtt{out}(\langle \mathtt{pk}(sk_S), g^{b_i}, \mathtt{sign}(sid, sk_S)\rangle)$
    $\mathtt{in}(\mathtt{enc}(sign, k))$
    if $\mathtt{verify}(sign, \mathtt{pk}(sk_P)) = sid$ then
      $\_S$.

Figure 9: Divided SSH Key Exchange

$$P_i^0; P_i^1(x_B, k)[R^P(k)] \| S_i^0; S_i^1(sid, k)[R^S(k)] \cong P_i^0; P_i^1(x_B, k)[Q^P(k)] \| S_i^0; S_i^1(sid, k)[Q^S(k)]$$

Without specifying the continuation, a first step toward the security of the basic SSH key exchange is to obtain Conditions K-1 and K-3 of Section 8. Recall that if a key exchange satisfies those Conditions, it can be seen as a secure key exchange in the classical sense as it can be composed with any continuation that do not share any long term secrets. The proofs only need to ne adapted when it is not the case.

The behaviour of the protocol is very similar to the signed DDH key exchange (Figure 4) previously studied. We can once again see the DH shares $\{a_i, b_i\}_{i\in\mathbb{N}}$ as local session identifiers that can be used to pair sessions. For each session and each party, the messages signed by this party always depend strongly on the DH share. We can thus make all SSH sessions simulatable with the following tagging functions and corresponding signing oracles.

$$\begin{array}{rl} T_P(m, \overline{s}) := & \exists s \in \{a_i\}_{i\in\mathbb{N}}, \exists m_1, m = \mathtt{hash}(g^s, m_1, m_1^s) \\ T_S(m, \overline{s}) := & \exists s \in \{b_i\}_{i\in\mathbb{N}}, \exists m_1, m = \mathtt{hash}(m_1, g^s, m_1^s) \end{array}$$

We have that the set of axioms $Ax = \text{EUF-CMA}_{T_P, sk_P, \overline{s}} \wedge \text{EUF-CMA}_{T_S, sk_S, \overline{s}}$ is $\mathcal{O}^{\mathtt{sign}}_{T_P, sk_P, \overline{s}}, \mathcal{O}^{\mathtt{sign}}_{T_S, sk_Q, \overline{s}}, \mathcal{O}_{a_i, b_i}$ sound thanks to Proposition 27. We use those axioms to perform the proof of K-3, where the tagging essentially implies the authentication property. However, the proof must be slightly stronger, when we consider that the continuations P, Q are instantiated with a second round of SSH with a forwarding agent that uses the same long term secrets.

9.3 SSH with Forwarding Agent

For concision, we write FA for $ForwardAgent$, SF for $SForward$, and PD for $PDistant$. Let us consider an abstract continuation protocol, satisfying a security property of the form $R^P(k) \| R^S(k) \cong Q^P(k) \| Q^S(k)$ where k denotes a fresh name modelling an ideal key produced by a key exchange.

We once again assume that the agents are only willing to communicate with the honest identities, i.e., $\mathtt{pk}(sk_S)$ and $\mathtt{pk}(sk_P)$ are predefined in the processes. The goal is to prove the following equivalence.

$$\begin{array}{lll} \|^i & (P_i[FA(k)] & \cong \|^i \quad (P_i[FA(k)] \\ & \|S_i[PD(k); R^P(k_{PD})] & \quad\quad\; \|S_i[PD(k); Q^P(k_{PD})] \\ & \|SF[R^S(k_{SF})]) & \quad\quad\; \|SF[Q^S(k_{SF})]) \end{array}$$

It corresponds to the fact that we should have $R^P(k) \| R^S(k) \cong Q^P(k) \| Q^S(k)$, even if the ideal key k is replaced for each party by a key derived by a SSH key exchange (PD and SF) using an forwarding agent (FA) based on a previous SSH key exchange (P and S).

We apply twice the decomposition of Section 8, once to show the security of the first key exchange (as done in the previous paragraph), and that we can thus prove the security of the second key exchange using an ideal key derived instead of the one derive by the first exchange. The second application is then used to prove the security of this second key exchange.

First application The first application is performed with the following Conditions (corresponding to the one of Section 8), which allow to derive the desired conclusion.

K-3):

$$\begin{array}{lll} P_0^0; & \mathsf{if}\ x_B \notin \overline{s}\ \mathsf{then} & \\ & \quad P_0^1(x_B,k); \mathtt{out}(k) & \\ & \mathsf{else}\ \mathtt{out}(k, g^{a_0}, x_B) & \\ \| S_0^0; & \mathsf{if}\ x_A \notin \overline{s}\ \mathsf{then} & \\ & \quad S_i^1(x_A,k); \mathtt{out}(k) & \\ & \mathsf{else}\ \mathtt{out}(k, g^{b_0}, x_A) & \end{array} \cong_{\mathcal{O}_{PS},\mathcal{O}_{forward}} \begin{array}{lll} P_0^0; & \mathsf{if}\ x_B = g^{b_0}\ \mathsf{then} & \\ & \quad \mathtt{out}(k, g^{a_0}, x_B) & \\ & \mathsf{else\ if}\ x_B \notin \overline{s}\ \mathsf{then} & \\ & \quad P^1(x_B,k); bad & \\ & \mathsf{else}\ \mathtt{out}(k, g^{b_0}, x_A) & \\ \| S_0^0; & \mathsf{if}\ x_A = g^{a_0}\ \mathsf{then} & \\ & \quad \mathtt{out}(k, g^{b_0}, x_A) & \\ & \mathsf{else\ if}\ x_B \notin \overline{s}\ \mathsf{then} & \\ & \quad S_0^1(x_A,k); bad & \\ & \mathsf{else}\ \mathtt{out}(k, g^{b_0}, x_A) & \end{array}$$

P-1):

$$\|^i P_i^1(k)[FA(k)] \| S_i^1(k)[PD(k); R^P] \| SF[R^S] \cong_{\mathcal{O}_{KE_1}} \|^i P_i^1(k)[FA(k)] \| S_i^1(k)[PD(k); Q^P] \| SF[Q^S]$$

We use the following oracles:

- $\mathcal{O}_{PS}$ allows to simulate (K-1) the other honest sessions of P and S, it corresponds to $\mathcal{O}^{\mathtt{sign}}_{T_P,F,sk_S,\overline{s}}, \mathcal{O}^{\mathtt{sign}}_{T_S,F,sk_P,\overline{s}}, \mathcal{O}_{a_i,b_i}$ of Section 9.2.
- $\mathcal{O}_{forward}$ allows to simulate (S-1) the continuation, i.e., protocols of the form $\mathtt{in}(k); P^1(k)[FA(k)] \| \mathtt{in}(k); S^1(k)[PD(k); R^P] \| SF[R^Q]$
- $\mathcal{O}_{KE_1}$ allows to simulate (S-2) $\|^i(P_i \| S_i)$ (it is identical to $\mathcal{O}_{PS}$).

All simulations are performed under $\nu sk_S, sk_P$. To define $\mathcal{O}_{forward}$, we need to settle an issue. Indeed, for hypothesis S-1, we need to provide an oracle that can simulate sessions of the forwarding protocols. However, in order to get the simulatability of $\mathtt{in}(k).FA(sk_P,k)$, one must give a generic signing oracles to the attacker, which would obviously make the protocol insecure. Based on the assumption that the forwarded sessions perform signatures tagged with "*fwd*" (as shown below), we can however provide a signing oracle for such messages only. It allows for the simulatability of the forwarding agent and of the forwarded client and server. More specifically, recall the the forwarding agent is of the form:

$$\begin{array}{l} FA(sk_P,k) := \\ \quad \mathtt{in}(\mathtt{enc}(sid,k)); \\ \quad\quad \mathtt{out}(\mathtt{enc}(\mathtt{sign}(\langle sid, \text{“}fwd\text{”}\rangle, sk_P), k)) \end{array}$$

We may obtain its simulatability with the following tagging function:

$$T_{for}(m,\overline{s}) := \quad \exists m_1.\ m = \langle m_1, \text{“}fwd\text{”}\rangle$$

Then, $\mathcal{O}_{forward}$ is simply $\mathcal{O}^{\mathtt{sign}}_{T_{for},F,sk_P,\overline{s}}, \mathcal{O}^{\mathtt{sign}}_{T_{for},F,sk_S,\overline{s}}, \mathcal{O}_{a'_i,b'_i}$. We prove Condition K-3 under the corresponding EUF-CMA axioms in Appendix E.

Second application We further simplify Condition P-1 of the previous paragraph with a second application of the decomposition of Section 8. We now denote $\overline{s}' = \{a'_i, b'_i\}_{i\in\mathbb{N}}$. PD_i and SF_i are split into PD_i^0, PD_i^1 and SF_i^0, SF_i^1 similarly to the split of Figure 9 before and after the key confirmation. The tagging functions used are only slight variations of the tagging functions for the first SSH key exchange:

$$\begin{array}{rl} T'_P(m,\overline{s}') := & \exists i, \exists X, m = \langle \mathtt{hash}(g^{a'_i}, X, X^{a'_i}), \text{“}fwd\text{”}\rangle \\ T'_S(m,\overline{s}') := & \exists i, \exists X, m = \langle \mathtt{hash}(X, g^{b'_i}, X^{b'_i}), \text{“}fwd\text{”}\rangle \end{array}$$

We then need to prove the Conditions:

K-3):

$$\begin{array}{rl} P_0^1(k); FA(k) \| S_0^1(k); PD_0^0(k); & \mathsf{if}\ x_B \notin \overline{s}'\ \mathsf{then} \\ & \quad PD_0^1(x_B,k); out(k) \\ & \mathsf{else}\ out(k, g^{a'_0}, x_B) \\ \| SF_0^0; & \mathsf{if}\ x_A \notin \overline{s}'\ \mathsf{then} \\ & \quad SF_0^1(x_A,k); out(k) \\ & \mathsf{else}\ out(k, g^{b'_0}, x_A) \end{array}$$

$$\cong_{\mathcal{O}_{KE_1}, \mathcal{O}^k_{FPS}, \mathcal{O}_{RQ}}$$

$$\begin{array}{rl} P_0^1(k); FA(k) \| S_0^1(k); PD_0^0; & \mathsf{if}\ x_B = g^{b'_0}\ \mathsf{then} \\ & \quad out(k, g^{a'_0}, x_B) \\ & \mathsf{else\ if}\ x_B \notin \overline{s}'\ \mathsf{then} \\ & \quad PD_0^1(x_B,k); bad \\ & \mathsf{else}\ out(k, g^{b'_0}, x_A) \\ \| SF_0^0; & \mathsf{if}\ x_A = g^{a'_0}\ \mathsf{then} \\ & \quad out(k, g^{b'_0}, x_A) \\ & \mathsf{else\ if}\ x_B \notin \overline{s}'\ \mathsf{then} \\ & \quad SF_0^1(x_A,k); bad \\ & \mathsf{else}\ out(k, g^{b'_0}, x_A) \end{array}$$

Note that k is a fresh name that could be considered as a long term secret, i.e., in $\overline{p}$.

And P-1):

$$\|^i PD_i^1(k'); R^P(k') \| SF_0^1(k'); R^S(k') \cong_{\mathcal{O}_{KE_1}, \mathcal{O}_{FPS}} \|^i PD_i^1(k'); Q^P(k') \| SF_i^1(k'); Q^S(k')$$

With the oracles:

- $\mathcal{O}^k_{FPS}$ allows to simulate (K-1) the other honest sessions of PD and SF, it corresponds to $\mathcal{O}^{\mathtt{sign}}_{T'_P, sk_S, \overline{s}}, \mathcal{O}^{\mathtt{sign}}_{T'_S, sk_P, \overline{s}}, \mathcal{O}_{a'_i, b'_i}$ of Section 9.2.
- $\mathcal{O}_{RQ}$ allows to simulate (S-1) the continuation, i.e., protocols of the form

$$\mathtt{in}(k); PD^1(k); R^P(k) \| \mathtt{in}(k); SF^1(k); R^Q(k)$$

We prove Condition K-3 under the corresponding EUF-CMA axioms in Appendix E. Remark that to ensure that the forwarding agent only signs the *sid* sent by PD, it is required that the encryption scheme is an authenticated encryption scheme.

Part III
Composition in the CCSA logic

10 Oracles in the CCSA Logic

We extend the semantics of the CCSA logic so that it now refers to attackers that can have access to an extra oracle $\mathcal{O}$. We then lift the notion of soundness for the axioms to support oracles, defining the notion of $\mathcal{O}$-soundness.

10.1 Syntax and Semantics

While the cryptographic library of the CCSA logic stays as is, the computational model must now also depend on some oracle that is given to the attacker, and the corresponding random oracle tape.

Definition 19. A *computational model* $\mathcal{M}$ is an extension of a cryptographic library $\mathcal{M}_f$, which provides an oracle $\mathcal{O}$, and an additional PTOM $\mathcal{A}_g^{\mathcal{O}}$ for each symbol $g \in \mathcal{G}$, that takes as input an infinite random tape ρ_r, a security parameter 1^η and a sequence of bitstrings.

We define the interpretation of extended terms as, given $\mathcal{M}$, η, σ, ρ_s , $\rho_{\mathcal{O}}$ and ρ_r:

- $[\![n]\!]^{\eta,\sigma}_{\mathcal{M},\rho_s,\rho_r,\rho_{\mathcal{O}}} := \mathcal{A}_n(1^\eta, \rho_s)$ if $n \in \mathcal{N}$
- $[\![x]\!]^{\eta,\sigma}_{\mathcal{M},\rho_s,\rho_r,\rho_{\mathcal{O}}} = [\![x\sigma]\!]^{\eta,\sigma}_{\mathcal{M},\rho_s,\rho_r,\rho_{\mathcal{O}}}$ if $x \in \mathcal{X}$
- $[\![f(\overline{u})]\!]^{\eta,\sigma}_{\mathcal{M},\rho_s,\rho_r,\rho_{\mathcal{O}}} = \mathcal{A}_f(1^\eta, [\![\overline{u}]\!]^{\eta,\sigma}_{\mathcal{M},\rho_s,\rho_r,\rho_{\mathcal{O}}})$ if $f \in \Sigma$
- $[\![g(\overline{u})]\!]^{\eta,\sigma}_{\mathcal{M},\rho_s,\rho_r,\rho_{\mathcal{O}}} = \mathcal{A}_g^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}([\![\overline{u}]\!]^{\eta,\sigma}_{\mathcal{M},\rho_s,\rho_r,\rho_{\mathcal{O}}}, \rho_r, 1^\eta)$ if $g \in \mathcal{G}$

We also adapt the definition of the interpretation of $\sim$.

Definition 20. Given a computational model $\mathcal{M}$, including an oracle $\mathcal{O}$, two sequences of terms $\overline{t}$, $\overline{u}$, and an assignment σ of the free variables of $\overline{t}, \overline{u}$ to ground terms, we have $\mathcal{M}, \sigma \models_{\mathcal{O}} \overline{t} \sim \overline{u}$ if, for every polynomial time oracle Turing machine $\mathcal{A}^{\mathcal{O}}$,

$$\begin{aligned}|&\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}([\![\overline{t}]\!]^{\sigma,\eta}_{\rho_s;\rho_r;\rho_{\mathcal{O}}}, \rho_r, 1^\eta) = 1\} \\ &-\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}([\![\overline{u}]\!]^{\sigma,\eta}_{\rho_s;\rho_r;\rho_{\mathcal{O}}}, \rho_r, 1^\eta) = 1\}|\end{aligned}$$

is negligible in η. Here, $\rho_s, \rho_r, \rho_{\mathcal{O}}$ are drawn according to a distribution such that every finite prefix is uniformly sampled.

10.2 Oracle Soundness

To perform proofs in the logic, we need to design axioms that are sound w.r.t. an attacker that has access to $\mathcal{O}$; we say that the axiom is $\mathcal{O}$-sound in this case. They should be easy to verify for actual libraries, yet powerful enough for the proofs that we intend to complete. The purpose of this Section is to provide such axioms. We first extend the notion of soundness to oracles.

Definition 21. Given a family of computational models $\mathcal{F}$ using oracle $\mathcal{O}$, a set of first order formulas A is $\mathcal{O}$-*sound* (w.r.t. $\mathcal{F}$) if, for every $\psi \in A$, every $\mathcal{M} \in \mathcal{F}$, $\mathcal{M} \models_{\mathcal{O}} \psi$.

With such a definition, if A is $\mathcal{O}$-sound (w.r.t. $\mathcal{F}$) and $A \models \phi$ (where ϕ is a closed formula), then, for every $\mathcal{M} \in \mathcal{F}$, $\mathcal{M} \models_{\mathcal{O}} \phi$.

Example 10.1 (Function application)**.** For any $\mathcal{O}$, $\mathcal{F}$, function f, terms $t_1, \ldots, t_n$, $u_1, \ldots, u_n$

$$t_1, \ldots, t_n \sim u_1, \ldots, u_n \implies f(t_1, \ldots, t_n) \sim f(u_1, \ldots, u_n)$$

is $\mathcal{O}$ sound.

Example 10.2. Given a single key encryption oracle $\mathcal{O}$ for key k, the formula

$$\mathtt{enc}(0, r, k) \sim \mathtt{enc}(1, r, k)$$

is

- not sound (nor $\mathcal{O}$-sound) in general,
- sound but not $\mathcal{O}$-sound for non randomized SPRP encryption,
- $\mathcal{O}$-sound for IND-CPA encryption.

Note that the axioms that are designed in [11] cannot be borrowed directly. For instance, $n \sim n'$, where n, n' are names, is a standard axiom: two randomly generated numbers of the same length cannot be distinguished. However, if either n or n' is in the support of $\mathcal{O}$, some information on their interpretation can be leaked by the oracle. The axiom $n \sim n'$ is sound, but not $\mathcal{O}$-sound. We have to modify this axioms as follows:

Lemma 22. *For any oracle $\mathcal{O}$ with support $\overline{n}$, the axiom $\forall k, k' \notin \overline{n}, k \sim k'$ is $\mathcal{O}$-sound.*

Proof. We are given a cryptographic library, and oracle $\mathcal{O}$ with support $\overline{n}$, and two names k, k' not in the support. We are also given $\mathcal{A}_{\mathcal{O}}$ which is a distinguisher over $k \sim k'$. We define a PTTM $\mathcal{A}'$ which on input $(m, \rho_r, 1^\eta)$:

- Splits ρ_r into three distinct infinite tapes ρ_{so}, ρ_{ra}, ρ_{ro}.
- Simulates $\mathcal{A}^{\mathcal{O}(\rho_{so}, \rho_{ro})}(m, \rho_{ra}, 1^\eta)$.

Let us a prove that $\mathcal{A}'$ is a distinguisher over $k \sim k'$, which contradicts the unconditional soundness of this axiom when there is no oracle.

We denote by $\pi_{\overline{k}}(\rho_s, \eta)$ the tapes where every bit of ρ_s which does not correspond to a name of $\overline{k}$ is set to 0, and similarly $\pi_{\overline{k}^c}(\rho_s, \eta)$ where all bits for $\overline{k}$ are set to 0. We then have for any PTOM $\mathcal{A}_{\mathcal{O}}$:

$$\begin{aligned}
&\mathbb{P}_{\rho_s, \rho_r, \rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s, \rho_{\mathcal{O}})}([\![\overline{k}]\!]^{\sigma,\eta}_{\rho_s}, \rho_r, 1^\eta) = 1\} \\
&=^1 \mathbb{P}_{\rho_s, \rho_r, \rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\pi_{\overline{k}}(\rho_s,\eta), \rho_{\mathcal{O}})}([\![\overline{n}]\!]^{\sigma,\eta}_{\pi_{\overline{k}^c}(\rho_s,\eta)}, \rho_r, 1^\eta) = 1\} \\
&=^2 \mathbb{P}_{\rho_{s1}, \rho_{s2}, \rho_r, \rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_{s1}, \rho_{\mathcal{O}})}([\![\overline{n}]\!]^{\sigma,\eta}_{\rho_{s2}}, \rho_r, 1^\eta) = 1\} \\
&=^3 \mathbb{P}_{\rho_{so}, \rho_s, \rho_{ra}, \rho_{ro}}\{\mathcal{A}^{\mathcal{O}(\rho_{so}, \rho_{ro})}([\![\overline{k}]\!]^{\sigma,\eta}_{\rho_s}, \rho_{ra}, 1^\eta) = 1\} \\
&=^4 \mathbb{P}_{\rho_s, \rho_r}\{\mathcal{A}'([\![\overline{k}]\!]^{\sigma,\eta}_{\rho_s}, \rho_r, 1^\eta) = 1\}
\end{aligned}$$

1. Thanks to the definition of support, the oracle answers the same on $\pi_{\overline{k}}(\rho_s, \eta)$ and ρ_s;
2. we split ρ_s in two, to replace independent tapes $\pi_{\overline{k}}(\rho_s, \eta)$ and $\pi_{\overline{k}^c}(\rho_s, \eta)$;

3. we rename random tapes;

4. by construction of $\mathcal{A}'$.

This shows that $\mathcal{A}'$ has the same advantage as $\mathcal{A}_{\mathcal{O}}$ against $k \sim k'$, which concludes the proof. □

Other axioms in [11] can be extended without problem. For instance the transitivity of $\sim$ or the function application axiom:

Lemma 23. *For any $\mathcal{O}$, $f \in \mathcal{F}$, terms $t_1, \ldots, t_n$, $u_1, \ldots, u_n$*

$$t_1, \ldots, t_n \sim u_1, \ldots, u_n \implies f(t_1, \ldots, t_n) \sim f(u_1, \ldots, u_n)$$

is $\mathcal{O}$ sound.

In general, what we have is that any axiom independent from the oracle support is sound.

Lemma 24. *For any $\mathcal{O}$, and terms t, s, such that all names in t, s do not appear in $\mathsf{supp}(\mathcal{O})$, we have that $t \sim s$ is sound if and only if $t \sim s$ is $\mathcal{O}$-sound.*

This allows us to derive, given an oracle and a recursive set of axiom, the set of axioms which is sound w.r.t. an oracle.

For instance, the general DDH axiom is, for any names a, b, c, $g^a, g^b, g^{ab} \sim g^a, g^b, g^c$. If we denote by $\overline{s}$ the support of some oracle, the $\mathcal{O}$-sound DDH version is simply the set of formulas $DDH_{\overline{s}}$ for all name $a, b, c \notin \overline{s}$, $g^a, g^b, g^{ab} \sim g^a, g^b, g^c$. Here, the notation g^x corresponds to $g(n)^{r(x)}$, where g is the function which extracts a group generator and r the function which evaluates names into exponents. We may consider that we have two interpretations of those function such that DDH holds.

EUF-CMA We define a CCSA version of the tagged EUF-CMA axiom. It is a direct adaptation of the CCSA EUF-CMA axiom to match the behaviour of the tagged EUF-CMA axiom (Figure 3).

Definition 25. Given a name sk and a function symbol T, we define the generic axiom scheme EUF-CMA$_{T,sk}$ as, for any term t such that sk is only in key position:

$$\begin{array}{l}\mathsf{if}\ (\ \mathtt{checksign}(t, pk(sk)))\\ \quad \mathsf{then}\ T(\ \mathtt{getmess}(t))\\ \qquad \dot{\bigvee}_{\mathtt{sign}(x,sk)\in \mathtt{St}(t)}\ (t \doteq \mathtt{sign}(x, sk))\\ \quad \mathsf{else}\ \top\end{array} \quad \sim\ \top$$

The tagged signing oracles is defined as previously, only adding the extra argument to the tagging function.

Definition 26. Given a name sk and a function T, we define the generic signing oracle $\mathcal{O}^{sign}_{T,sk}$ as follows:

$$\mathcal{O}^{\mathtt{sign}}_{T,sk}(m) := \mathsf{if}\ T(m)\ \mathsf{then}\ \ \mathrm{output}(\mathtt{sign}(m, sk)))$$

Proposition 27. *For any computational model in which the interpretation of* $\mathtt{sign}$ *is* EUF-CMA*, any name* sk*, and any boolean function* T*,* EUF-CMA$_{T,sk}$ *is* $\mathcal{O}^{\mathtt{sign}}_{T,sk}$*-sound.*

Proof. Let us assume that soundness is violated. We then have a term t and a computational model such that t does not satisfy EUF-CMA$_{T,sk}$. It means that the formula on the left hand side holds. As in t the secret key sk only occurs in key positions, we can simulate t by sampling all names, performing applications of function symbols, and sometimes calling the oracle $\mathcal{O}^{\mathtt{sign}}_{sk}$ to obtain a signature. t may also depend on attacker function symbols that have access to an oracle $\mathcal{O}^{\mathtt{sign}}_{T,sk}$. Thus, we can build a PTOM $\mathcal{A}^{\mathcal{O}^{\mathtt{sign}}_{T,sk},\mathcal{O}^{\mathtt{sign}}_{sk}}$ that produces exactly the same distribution of t for any fixed value of sk.

Let $\mathcal{B}^{\mathcal{O}^{\mathtt{sign}}_{sk}}$ be the PTOM which:

- simulates $\mathcal{A}^{\mathcal{O}^{\mathtt{sign}}_{T,sk}}$, by sampling all names itself, except sk;
- for every call made by A to $\mathcal{O}^{\mathtt{sign}}_{T,sk}$ with input m, $\mathcal{B}$ checks that $T(M)$ holds, and if it is the case query the signing oracle to get the signature, else fails.

The probability distribution of $\mathcal{B}^{\mathcal{O}^{\mathtt{sign}}_{sk}}$ is exactly the same as $\mathcal{A}^{\mathcal{O}^{\mathtt{sign}}_{T,sk},\mathcal{O}^{\mathtt{sign}}_{sk}}$, so $\mathcal{B}^{\mathcal{O}^{\mathtt{sign}}_{sk}}$ also produces an output o which violates the EUF-CMA$_{T,sk}$ axiom. We thus have that o is a valid signature, and is either not well tagged or does not correspond to a sub-term of t.

As all calls to $\mathcal{O}^{\mathtt{sign}}_{sk}$ made by $\mathcal{B}$ either correspond to a well tagged message or to a sub term of t, we know that o does not correspond to a signature produced by the signing oracle. $\mathcal{B}^{\mathcal{O}^{\mathtt{sign}}_{sk}}$ is thus an attacker which given access to a signing oracle can produce a signature for a message not signed by the oracle, i.e., an attacker which can win the EUF-CMA axiom. □

11 Computational Soundness of the logic

11.1 Protocols

Given a protocol P, we reuse from [11] the definition of $\Phi(\mathbf{fold}(P))$ which we will denote t_P. It is only needed for technical proofs. We remark here that with the notations of [11], we would have $\rho_1 = \rho_s$ and $\rho_2 = \rho_r$.

The correction of the term representing a protocol with respect to the protocol oracles is given by the following Lemma.

Lemma 28. *Given a protocol* P *(which is action deterministic), a functional model* $\mathcal{M}^f$*, an oracle* $\mathcal{O}$*, a security parameter* $\eta \in \mathbb{N}$*, an history tape* $\theta = \emptyset$*,* $t_P = t^1_P, \ldots, t^n_P$*,* $\sigma := \{x_1 \mapsto d_1, \ldots, x_n \mapsto d_n\}$ *an assignment of the free variables in* t_P *to* D*, for every* $\rho_s, \rho_r, \rho_{\mathcal{O}}$*,*

$$[\![t^1_P]\!]^{\sigma,\eta}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}, \ldots, [\![t^n_P]\!]^{\sigma,\eta}_{\rho_s,\rho_r,\rho_{\mathcal{O}}} =$$
$$\mathcal{O}_P(\rho_s,\emptyset)(d_1(\rho_s,\rho_r,\eta,\rho_{\mathcal{O}})), \ldots, \mathcal{O}_P(\rho_s, d_1(\rho_s,\rho_r,\eta,\rho_{\mathcal{O}}), \ldots, d_{n-1}(\rho_s,\rho_r,\eta,\rho_{\mathcal{O}}))(d_n(\rho_s,\rho_r,\eta,\rho_{\mathcal{O}}))$$

Proof. While straightforward, the proof relies on the definitions of protocol execution in a model defined in [11] and the soundness of the folding, which we do not recall here. We extend $\mathcal{M}^f$ into a computational model $\mathcal{M}$ in such a way that

$$[\![g_i]\!]([\![t^1_P]\!]^{\sigma,\eta}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}, \ldots, [\![t^{i-1}_P]\!]^{\sigma,\eta}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}, \rho_r) = d_i(\rho_s,\rho_r,\eta,\rho_{\mathcal{O}})$$

for $i = 1, \ldots, n$. We then have $[\![t_P]\!]_{\mathcal{M}} = [\![t^1_P]\!]^{\sigma,\eta}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}, \ldots, [\![t^n_P]\!]^{\sigma,\eta}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}$.

The folding soundness from [11] implies that $P \sim_{\mathcal{M}} \mathbf{fold}(P)$. The proof actually implies pointwise equality of the executions of P and $\mathbf{fold}(P)$ in $\mathcal{M}$. If we denote $\psi(P)$ (resp $\psi(\mathbf{fold}(P))$) the sequence of outputs of the execution of P (resp $\mathbf{fold}(P)$) in this model, we thus have that $\psi(P) = \psi(\mathbf{fold}(P))$.

We directly have by definition of the t_P that $[\![t_P]\!]_{\mathcal{M}} = [\![\Phi(\mathbf{fold}(P)]\!]_{\mathcal{M}} = \psi(\mathbf{fold}(P))$. Finally, by construction of $\mathcal{O}_P$ which emulates exactly the execution of P we have $\psi(P) = \mathcal{O}_P(\rho_s, \emptyset)(d_1(\rho_s, \rho_r, \eta, \rho_{\mathcal{O}})), \ldots, \mathcal{O}_P(\rho_s, d_1(\rho_s, \rho_r, \eta, \rho_{\mathcal{O}}), \ldots, d_{n-1}(\rho_s, \rho_r, \eta, \rho_{\mathcal{O}}))(d_n(\rho_s, \rho_r, \eta, \rho_{\mathcal{O}}))$ which concludes the proof. □

11.2 Introduction of attacker's functions

As in [11], we may replace the variables occurring in the protocol P (or its folding t_P) with terms that include the attacker functions $g \in \mathcal{G}$.

If $t_P = t_P^0, \ldots, t_P^n$, we let $\widetilde{t_P} = \widetilde{t_P^0}, \ldots, \widetilde{t_P^n}$ be the sequence of terms defined by:

- $\widetilde{t_P^0} = t_P^0\{x_0 \mapsto g_0()\}$ and $\phi_0^P = \emptyset$
- $\widetilde{t_P^{i+1}} = t_P^{i+1}\{x_0 \mapsto g_0(), x_1 \mapsto g_1(\phi_1^P), \ldots, x_{i+1} \mapsto g_{i+1}(\phi_{i+1}^P)\}$ and $\phi_{i+1}^P = \phi_i^P, \widetilde{t_P^{i+1}}$

We then denote σ_P the substitution $\{x_0 \mapsto g_0(), x_1 \mapsto g_1(\phi_1^P), \ldots, x_n \mapsto g_n(\phi_n^P)\}$.

There is exactly one attacker function for every message produced by the protocol, and the function symbol are defined independently from the protocol.The functions are placeholder for the attacker actions, whom we give the previous answers he may have obtained in the protocol.

Example 11.1. We consider the protocol which for a given key sk, will allow the attacker to perform one decryption and will then output an encryption. We may have $t_P = dec(x, sk), enc(y, r, sk)$, where x and y are the two expected inputs. Then $\widetilde{t_P} = dec(g_0()), sk), enc(g_1(dec(g_0()), sk)$ When we interpret this term, the attacker can choose the evaluation of g_0 and g_1. He can at first provide the protocol with a message and obtain its decryption, and can then compute a new message, maybe based on the previous decryption he obtained.

Once we have fixed the cryptographic library, and we consider two protocols P and Q, a computational such that $\widetilde{t_P} \not\sim \widetilde{t_Q}$ means that we have multiple PPTOMs which can compute messages so that in the end, a final PPTOM can distinguish the two protocols. We may from those machines reconstruct a single machine, which is an attacker against $P \cong_{\mathcal{O}} Q$. Conversely, an attacker against $P \cong_{\mathcal{O}} Q$ may be split into multiple machines, so that a machine computes the next message given by the attacker to the protocol, those machines providing in the end a computational model such that $\widetilde{t_P} \not\sim \widetilde{t_Q}$.

Formally, we have the computational soundness of our oracle indistinguishability.

Lemma 29. *Given two protocols P, Q, random tapes ρ_r, ρ_s, a cryptographic library $\mathcal{M}_f$ and an oracle $\mathcal{O}$, we have:*

$$\forall \mathcal{M} \supset \mathcal{M}_f. \quad \mathcal{M} \models_{\mathcal{O}} \widetilde{t_P} \sim \widetilde{t_Q}$$

$$\Leftrightarrow$$

$$P \cong_{\mathcal{O}} Q$$

We finally have a result of computational soundness. We write $Ax \models \phi$ if the set of formulas Ax and the formula $\neg\phi$ are inconsistent.

Theorem 6. *Given P, Q two protocols, $\mathcal{O}$ an oracle, A a set of axioms ,$\mathcal{M}^f$ a cryptographic library we assume that:*

- *A is $\mathcal{O}$-sound w.r.t $\mathcal{F} = \{\mathcal{M} \supset \mathcal{M}^f\}$*
- *$A \models \widetilde{t_P} \sim \widetilde{t_Q}$*

Then $P \cong_{\mathcal{O}} Q$

Proof. Let us assume that we have a distinguisher on $\mathcal{A}^{\mathcal{O},\mathcal{O}_{P?Q}}$ and that A is $\mathcal{O}$-sound.

With Lemma 29 we have a computational model $\mathcal{M} \supset \mathcal{M}^f$ such that $\mathcal{M} \models_{\mathcal{O}} \widetilde{t_P} \not\sim \widetilde{t_Q}$. As A is $\mathcal{O}$-sound, we also have $\mathcal{M} \models_{\mathcal{O}} A$, and this contradicts the fact that the formulas are inconsistent. □

We reduce computational indistinguishability to an inconsistency proof on the one hand and a soundness proof of the axioms on the other hand.

12 Extension to the Model for Unbounded Replication

Recall that for unbounded replications, we used notations such as $x \notin \overline{s}$, for infinite sequences of names $\overline{s}$. While the previous extension is enough to handle our composition results, we need for our applications to key exchanges to be able to express formally those predicates. To this end, for any name n of arity l, we give a formal interpretation to $\overline{n}$, that intuitively models the sequence of names $n_{1,\ldots,1}, \ldots, n_{r_1,\ldots,r_l}$ of length polynomial in the security parameter.

We define the syntax and provide variations of the axioms that can be used to reason in this context. We then provide the concrete semantics so that these axioms are sound as technical details.

We provide a way to support infinite sequences in the CCSA logic, but note that our composition framework does not always require infinite sequences. When considering basic key exchanges, it is enough to use cofinite sequences. Basically, if the property

$$KE_0[\mathsf{if}\ x^I_{lsid} = lsid^R_0\ \mathsf{then}\ \mathtt{out}(k)\ \mathsf{else}\ \mathtt{out}(x^I_0), \mathsf{if}\ x^R_{lsid} = lsid^I_0\ \mathsf{then}\ \mathtt{out}(k)\ \mathsf{else}\ \mathtt{out}(x^R_0)]$$

holds even when the attacker can simulate corrupted sessions, it is enough to derive the security of multiple sessions. It is interesting, as this property does not rely on infinite sequences.

To understand this, let us briefly consider a basic unsigned Diffie Hellman key exchange. It must of course not verify the previous property. The exchange shares are g^{a_0}, g^{b_0}. To break the previous property, we can give as a share to I the correct g^{b_0}, I will then produce depending on the side k or $g^{a_0 b_0}$. If we provide R with $g^{a_0} \times g^{a_0}$, R does not believe to be paired with I and it then always output as key $g^{2a_0 b_0}$. One can then easily distinguish if the output of R is the square of the output of I.

Basically, this stems from the fact that always outputting the actual key leaks information to the attacker when agents are not paired together.

For key exchanges with key confirmation, we wish to test the real or random before we have any authentication (as the authentication may come from the key confirmation). So if we always leak the key of the agent, the property will not be verified. However, we do need to leak the key to enable to go from one session to multiple sessions (to give the attacker enough information for the simulatability). The idea is then, as expressed in the previous Theorems,

to only leak the key when two "honest" parties are paired together. Else, we execute the key confirmation, which should fail. Here, we have an explicit need to be able to test which sessions are honest, whether they are corrupted or not, and this for an unbounded number of sessions. Hence the need for a test based on infinite sequences.

Syntax Recall that names are defined with an arity, where a name n of index arity l can be indexed by l integers, yielding a distinct copy of the name for each indexes. Moreover, in a protocol, the index variables occurring in names must all be bound through a parallel or a sequential binder, and thus once we consider the term corresponding to the protocol in the CCSA logic, all names appear without index variables.

For any name n of index arity l, the syntax of terms in the CCSA logic only contained all the copies $n_{k_1,\dots,k_l}$ for $k_1,\dots,k_l \in \mathbb{N}$ as symbols of arity 0 (a constants of the term algebra). For each name n, we add to the syntax of terms the symbol seq_n of arity 0. We also provide a function symbol $\in$ using infix notation, so that $t \in \mathsf{seq}_n$ is now in the syntax.

Axioms The classical α-renaming axiom still holds, but all copies of a name are renamed at once. Thus, for any sequences of terms $\overline{t}$, and any names n, n' of index arity l such that n' does not occur in t, we have:

$$(1)\ \overline{t} \sim \overline{t}\{\mathsf{seq}_n \mapsto \mathsf{seq}_{n'}\} \cup \{n_{k_1,\dots,k_l} \mapsto n'_{k_1,\dots,k_l} \mid k_1,\dots,k_l \in \mathbb{N}\}$$

Furthermore, we also provide axioms that allow to reason about the membership predicate, defined as:

(2) $n_{k_1,\dots,k_l} \in \mathsf{seq}_n \sim \mathtt{true}$ for any name n and all $k_1,\dots,k_l \in \mathbb{N}$;

(3) $n'_{k_1,\dots,k_l} \in \mathsf{seq}_n \sim \mathtt{false}$ for any name n' distinct of n and all $k_1,\dots,k_l \in \mathbb{N}$.

Remark that as $\in$ is a boolean function symbol, it is in contradiction with its negation and we trivially have that that for any term t and name n,

$$t \in \mathsf{seq}_n \wedge t \notin \mathsf{seq}_n \sim \mathtt{false}$$

This is actually what is used in our proofs of indistinguishability, as tagged oracles in our applications provide messages m such that we have $f(m) \in \mathsf{seq}_n$ for some function f, and the security property raises bad if $f(m) \notin \mathsf{seq}_n$.

Semantics The idea is that seq_n should model all sequences $\mathsf{seq}_n = \{n_1,\dots,n_{p(\eta)}\}$ for any polynomial p. Then, if an indistinguishability holds for all such sequences for all polynomials, it also holds when the polynomial is bigger than the running time of the distinguisher, and the sequence then models an infinite sequence. To model this, the interpretation of a term t may now depend on some polynomial p with one indeterminate and with positive integer coefficients given to the PTTMs, and the interpretation is denoted $[\![t]\!]^{\eta,\sigma}_{\mathcal{M},p,\rho_s,\rho_r,\rho_{\mathcal{O}}}$.

The indistinguishability predicate $\sim$ is now interpreted as indistinguishability for all distinguishers and all polynomials p. Definition 20 now becomes:

Definition 30. Given a computational model $\mathcal{M}$, including an oracle $\mathcal{O}$, two sequences of terms $\overline{t}$, $\overline{u}$, and an assignment σ of the free variables of $\overline{t}, \overline{u}$ to ground terms, we have $\mathcal{M}, \sigma \models_{\mathcal{O}}$

$\overline{t} \sim \overline{u}$ if, for any strictly increasing polynomial p and every polynomial time oracle Turing machine $\mathcal{A}^{\mathcal{O}}$,

$$|\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(p,\rho_s,\rho_{\mathcal{O}})}([\![\overline{t}]\!]^{\sigma,\eta}_{\mathcal{M},p,\rho_s;\rho_r;\rho_{\mathcal{O}}}, \rho_r, 1^\eta) = 1\} \\ -\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(p,\rho_s,\rho_{\mathcal{O}})}([\![\overline{u}]\!]^{\sigma,\eta}_{\mathcal{M},p,\rho_s;\rho_r;\rho_{\mathcal{O}}}, \rho_r, 1^\eta) = 1\}|$$

is negligible in η. Here, $\rho_s, \rho_r, \rho_{\mathcal{O}}$ are drawn according to a distribution such that every finite prefix is uniformly sampled.

So, we can now assume that the interpretation of terms may depend on a polynomial p. We previously assumed for a name n_i, that the cryptographic library was providing a distinct Turing Machine for each copy of the name, i.e., a machine $\mathcal{A}_{n_k}$ for each $k \in \mathbb{N}$. However, to build a machine that can interpret seq_n, all the copies of the name must be extracted in a uniform way, so that it is possible to collect all of them in polynomial time. To this end, we now consider that a cryptographic library provides, for each name $n_{\bar{i}}$ of index arity l, a Turing Machine $\mathcal{A}_n$ that takes as input the security parameter, the random tape ρ_S and l integers, and returns a sequence of bitstrings of length η extracted from ρ_s. Then, the interpretation of the name $n_{k_1,\ldots,k_l}$, with $k_1, \ldots, k_l \in \mathbb{N}$ is, given $\mathcal{M}$, η, σ, ρ_s , $\rho_{\mathcal{O}}$ and ρ_r.

$$[\![n_{k_1,\ldots,k_l}]\!]^{\eta,\sigma}_{\mathcal{M},p,\rho_s,\rho_r,\rho_{\mathcal{O}}} := \mathcal{A}_n(1^\eta, \rho_s, k_1, \ldots, k_l)$$

The set of all the $\mathcal{A}_n$ should use distinct parts of the random tape ρ_s, and each $\mathcal{A}_n$ should return distinct parts of the tape for each sequence of integers given as integers. This can be done for instance if ρ_s is seen as a folding of random tapes $\rho_{s,n}$ in a single tape, such that each $\mathcal{A}_n$ only accesses bits corresponds to $\rho_{s,n}$ through the inverse folding (this essentially corresponding to bijective mappings from $\mathbb{N}^k$ to $\mathbb{N}$). Then, for each sequence of integers $k_1, \ldots, k_l$, $\mathcal{A}_n$ extracts from $\rho_{s,n}$ a unique sequence of bits by computing a bijection f from $\mathbb{N}^l$ to $\mathbb{N}$, and extracting the bitstrings of length η at position $\eta \times f(k_1, \ldots, k_l)$.

Using this new interpretation for names, we now define the semantics of seq_n, for any name n of index arity l, as, given $\mathcal{M}$ (that now contains a polynomial p), η, σ, ρ_s, $\rho_{\mathcal{O}}$ and ρ_r,

$$[\![\mathsf{seq}_n]\!]^{\eta,\sigma}_{\mathcal{M},p,\rho_s,\rho_r,\rho_{\mathcal{O}}} := \mathcal{A}_{\mathsf{seq}_n}(1^\eta, p, \rho_s)$$

where $\mathcal{A}_{\mathsf{seq}_n}$ is the machine that:

- contains l nested loops over the l variables $c_1, \ldots, c_l$ all ranging from 1 to $p(\eta)$;
- at each iteration, simulate $\mathcal{A}_n(1^\eta, \rho_s, c_1, \ldots, c_l)$ and appends its result to the output tape.

Remark that given a model $\mathcal{M}$, and thus the machine $\mathcal{A}_n$, we completely fix the machine $\mathcal{A}_{\mathsf{seq}_n}$. Essentially, $\mathcal{A}_{\mathsf{seq}_n}$ will produce the sequence of bitstring corresponding to the interpretation of $n_{1,\ldots,1}, \ldots, n_{p(\eta),\ldots,p(\eta)}$.

The CCSA axioms presented previously are still sound in this semantics. Essentially, this is because when the axiom scheme does not depend on any seq_n, all the occurrences of seq_n in terms satisfying the guards of the scheme can be simulated by an attacker who samples $p(\eta)$ randoms.

Lemma 31. *For any computational model in which the interpretation of* $\mathtt{sign}$ *is* EUF-CMA*, any name sk,* EUF-CMA$_{T,sk}$ *is* $\mathcal{O}^{\mathtt{sign}}_{T,sk}$*-sound even for terms that may depend on some* seq_n*.*

Proof. We have a term t, a computational model and a polynomial p such that the interpretation of t where all sequences seq_n are of length $p(\eta)$ contradicts the EUF-CMA$_{T,sk}$ axiom.

The proof is exactly the same as Proposition 27, as we can once again from t build a Turing Machine that samples all names but sk (and may thus sample $p(\eta)$ names for each sequence), and is then able to simulate all operations of t. □

This means that we can safely consider a version of EUF-CMA$_{T,sk}$ where for instance $T(x)$ is of the form $x \in \mathsf{seq}_n$ and still have the soundness of the axiom. Remark that this proof would hold similarly for other cryptographic axioms.

We however have to prove the soundness of the axioms that are specific to seq.

Proposition 32. *Axioms (1),(2) and (3) are sound in all models where the interpretation of $\in$ is given by the machine $\mathcal{A}_\in(1^\eta, x_1, x_2)$ that checks if x_1 is a bitstring of length η and returns true if and only if x_1 is a sub-string of x_2 starting at a position which is a multiple of η.*

Proof.

1. The alpha-renaming axiom is sound, unconditionally. This is similar to the classical CCSA logic alpha-renaming axiom, which holds as all randomness for a given name (of any arity) are completely independent and uniform. Replacing all occurrences of a name by a another fresh one thus yields exactly the same distribution. In essence, we replace in the interpretation of $\bar{t}$ all occurrences of $\mathcal{A}_n$ and $\mathcal{A}_{\mathsf{seq}_n}$ by $\mathcal{A}_{n'}$ and $\mathcal{A}_{\mathsf{seq}_{n'}}$. As the machines for n' did not occur previously in the interpretation of $\bar{t}$, we indeed have that the machines of n and of n' produce the same independent distribution for the interpretation of $\bar{t}$.

2. Given $n_{k_1,\ldots,k_l}$ and seq_n, we have for any polynomial p strictly increasing that for η large enough, $k_i \leq p(\eta)$ for $1 \leq i \leq l$. Thus, for η large enough, the interpretation of seq_n contains the result of $\mathcal{A}_n(1^\eta, \rho_s, k_1, \ldots, k_l)$ (simulated by $\mathcal{A}_{\mathsf{seq}_n}$), and $\mathcal{A}_\in$ always output `true`. The advantage of any attacker then becomes 0 which is negligible.

3. The probability of collision between two sequences of bitstrings of length η is $\frac{1}{2^\eta}$. For any polynomial p, as seq_n is a uniform sampling of length $p(\eta) \times \eta$, and $n'_{k_1,\ldots,k_l}$ is an independent uniform sampling of length η, the probability that $n'_{k_1,\ldots,k_l}$ occurs in seq_n at a position which is a multiple of η is the probability $1 - (1 - \frac{1}{2^\eta})^{p(\eta)}$. Thus, $\mathcal{A}_\in$ will answer true with only a negligible probability.

□

As the interpretation $\mathcal{A}_\in$ given in the previous proposition corresponds to the interpretation required in the application to key exchanges (Section 6), we can indeed use those axioms in proofs of key exchange security.

References

[1] R. Canetti, *Universally Composable Security: A New Paradigm for Cryptographic Protocols*, 2000. [Online]. Available: http://eprint.iacr.org/2000/067

[2] R. Canetti and T. Rabin, "Universal Composition with Joint State," in *Advances in Cryptology - CRYPTO 2003*, ser. Lecture Notes in Computer Science, D. Boneh, Ed. Springer Berlin Heidelberg, 2003, pp. 265–281.

[3] M. Backes, B. Pfitzmann, and M. Waidner, "The Reactive Simulatability (RSIM) Framework for Asynchronous Systems," *Inf. Comput.*, vol. 205, no. 12, pp. 1685–1720, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.ic.2007.05.002

[4] D. Hofheinz and V. Shoup, "GNUC: A New Universal Composability Framework," *Journal of Cryptology*, vol. 28, no. 3, pp. 423–508, Jul. 2015. [Online]. Available: https://doi.org/10.1007/s00145-013-9160-y

[5] M. Backes, M. Dürmuth, D. Hofheinz, and R. Küsters, "Conditional reactive simulatability," *Int. J. Inf. Sec.*, vol. 7, no. 2, pp. 155–169, 2008.

[6] J. Camenisch, S. Krenn, R. Küsters, and D. Rausch, "iUC: Flexible Universal Composability Made Simple," Tech. Rep., 2019.

[7] U. Maurer, "Constructive cryptography - A new paradigm for security definitions and proofs," in *TOSCA*, ser. Lecture Notes in Computer Science, vol. 6993. Springer, 2011, pp. 33–56.

[8] C. Brzuska, M. Fischlin, N. P. Smart, B. Warinschi, and S. C. Williams, "Less is more: relaxed yet composable security notions for key exchange," *International Journal of Information Security*, vol. 12, no. 4, pp. 267–297, Aug. 2013. [Online]. Available: https://doi.org/10.1007/s10207-013-0192-y

[9] B. Blanchet, "Composition Theorems for CryptoVerif and Application to TLS 1.3," in *31st IEEE Computer Security Foundations Symposium (CSF'18)*. Oxford, UK: IEEE Computer Society, Jul. 2018, pp. 16–30.

[10] C. Brzuska, A. Delignat-Lavaud, C. Fournet, K. Kohbrok, and M. Kohlweiss, "State separation for code-based game-playing proofs," in *ASIACRYPT (3)*, ser. Lecture Notes in Computer Science, vol. 11274. Springer, 2018, pp. 222–249.

[11] G. Bana and H. Comon-Lundh, "A computationally complete symbolic attacker for equivalence properties," in *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS'14)*, G.-J. Ahn, M. Yung, and N. Li, Eds. Scottsdale, Arizona, USA: ACM Press, Nov. 2014, pp. 609–620. [Online]. Available: http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/BC-ccs14.pdf

[12] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol." [Online]. Available: https://tools.ietf.org/html/rfc4253

[13] R. Küsters and D. Rausch, "A Framework for Universally Composable Diffie-Hellman Key Exchange," in *IEEE 38th Symposium on Security and Privacy (S&P 2017)*. IEEE Computer Society, 2017, pp. 881–900.

[14] G. Scerri and S.-O. Ryan, "Analysis of Key Wrapping APIs: Generic Policies, Computational Security." IEEE Computer Society, Jun. 2016, pp. 281–295. [Online]. Available: https://hal.inria.fr/hal-01417123

[15] H. Comon and A. Koutsos, "Formal Computational Unlinkability Proofs of RFID Protocols," in *Proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF'17)*, B. Köpf and S. Chong, Eds. Santa Barbara, California, USA: IEEE Computer Society Press, Aug. 2017, pp. 100–114. [Online]. Available: http://ieeexplore.ieee.org/document/8049714/

[16] G. Bana, R. Chadha, and A. K. Eeralla, "Formal Analysis of Vote Privacy Using Computationally Complete Symbolic Attacker," in *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part II*, 2018, pp. 350–372. [Online]. Available: https://doi.org/10.1007/978-3-319-98989-1_18

[17] "ISO/IEC 9798-3:2019, IT Security techniques – Entity authentication – Part 3: Mechanisms using digital signature techniques." [Online]. Available: https://www.iso.org/standard/67115.html

[18] C. Brzuska, M. Fischlin, B. Warinschi, and S. C. Williams, "Composability of Bellare-rogaway Key Exchange Protocols," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 51–62. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046716

[19] M. Fischlin and F. Günther, "Multi-Stage Key Exchange and the Case of Google's QUIC Protocol," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 1193–1204, event-place: Scottsdale, Arizona, USA. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660308

[20] R. Küsters and M. Tuengerthal, "Composition Theorems Without Pre-established Session Identifiers," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 41–50, event-place: Chicago, Illinois, USA. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046715

[21] N. Durgin, J. Mitchell, and D. Pavlovic, "A Compositional Logic for Proving Security Properties of Protocols," *J. Comput. Secur.*, vol. 11, no. 4, pp. 677–721, Jul. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=959088.959095

[22] A. Datta, A. Derek, J. C. Mitchell, V. Shmatikov, and M. Turuani, "Probabilistic Polynomial-Time Semantics for a Protocol Security Logic," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, Eds. Springer Berlin Heidelberg, 2005, pp. 16–29.

[23] C. Cremers, "On the Protocol Composition Logic PCL," in *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '08. New York, NY, USA: ACM, 2008, pp. 66–76, event-place: Tokyo, Japan. [Online]. Available: http://doi.acm.org/10.1145/1368310.1368324

[24] S. C. Williams, "Analysis of the SSH Key Exchange Protocol," in *Cryptography and Coding*, ser. Lecture Notes in Computer Science, L. Chen, Ed. Springer Berlin Heidelberg, 2011, pp. 356–374.

[25] D. Cadé and B. Blanchet, "From Computationally-Proved Protocol Specifications to Implementations and Application to SSH," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 4, no. 1, pp. 4–31, Mar. 2013.

[26] B. Blanchet, "CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols," in *Dagstuhl seminar "Formal Protocol Verification Applied"*, Oct. 2007.

[27] V. Cortier and S. Delaune, "A method for proving observational equivalence," in *2009 22nd IEEE Computer Security Foundations Symposium*. IEEE, 2009, pp. 266–276.

[28] M. Arapinis, V. Cheval, and S. Delaune, "Verifying Privacy-Type Properties in a Modular Way," in *2012 IEEE 25th Computer Security Foundations Symposium*, Jun. 2012, pp. 95–109.

[29] K. G. Paterson, J. C. N. Schuldt, M. Stam, and S. Thomson, "On the Joint Security of Encryption and Signature, Revisited," in *Advances in Cryptology – ASIACRYPT 2011*, ser. Lecture Notes in Computer Science, D. H. Lee and X. Wang, Eds. Springer Berlin Heidelberg, 2011, pp. 161–178.

A Messages

Protocols and oracles produce messages depending on names, randomness and some cryptographic primitives. We define here formally a syntax and a semantic for such messages.

A.1 Syntax of messages

We build terms over $\mathcal{F}$ a set of function symbols, which will represent the honest function symbols (encryption symbol, decryption), a set of variables $\mathcal{X}$ (unknown terms), and a set of names $\mathcal{N}$, intended to denote respectively the secret and public names. Names may be sorted, for instance to capture what is a secret key and what is a randomness.

Example A.1. We define $\mathcal{F} := \{enc/3, dec/2\}$ an encryption scheme, $\mathcal{N} = \{k, r\}$ a secret key, a key and a randomness. Then, with $mess$ an arbitrary term $t_1 = enc(mess, k, r)$ represents the encryption of an arbitrary message, and $dec(t_1, k)$ represents its decryption.

A.2 Semantics of terms

We wish to describe protocols (i.e messages) as terms, whose interpretation must be fixed and deterministic. This allows us to obtain an interpretation, which is uniform, and we provide this interpretation with explicit randomness.

Messages will thus be interpreted as deterministic PPT, which takes as inputs:

- ρ_s, a random tape for secret names (e.g secret keys)
- 1^η, the security parameter

Let D be the set of such PPT, called *messages*.

A *cryptographic library* $\mathcal{M}_f$ is a mapping $[\![\cdot]\!]_{\mathcal{M}_f}$ that interprets the function symbols, names and closed terms in the set of messages. The index $\mathcal{M}_f$ is omitted unless there is some ambiguity, in order to avoid overloaded notations. $[\![\cdot]\!]_{\mathcal{M}}$ is defined as follows:

1. if $n \in \mathcal{N}$, n is interpreted as the machine $[\![n]\!]_{\mathcal{M}} = \mathcal{A}_n$ that on input $(1^\eta, \rho_s)$ extracts a word of length η from the tape ρ_s. Different names should extract disjoint parts of the random tape.

2. if $f \in \mathcal{F}$, then, with $d_1, ..., d_n \in D^n$ a sequence of messages, $[\![f]\!]_{\mathcal{M}}(d_1, \ldots, d_n)$ is the machine such that, on input $(1^\eta, \rho_s)$,

$$[\![f]\!]_{\mathcal{M}}(d_1, \ldots, d_n)(1^\eta, \rho_s) := \mathcal{A}_f(d_1(1^\eta, \rho_s, \rho_{\mathcal{O}}), \ldots, d_n(1^\eta, \rho_s))$$

   Intuitively, we simply compose the machine, which represents f, with all the machines representing its inputs. f can only be deterministic, any randomness must be explicitly given as an argument.

Given an assignment σ of variables to messages in D, the random tape ρ_s, and a security parameter $\eta \in \mathbb{N}$, for each $f \in \mathcal{F}$ a Turing machine A_f, the (evaluation of the) interpretation of a term t is inductively defined as follows:

- $[\![n]\!]^{\eta,\sigma}_{\rho_s} := \mathcal{A}_n(1^\eta, \rho_s)$ if $n \in \mathcal{N}$
- $[\![x]\!]^{\eta,\sigma}_{\rho_s} = (x\sigma)(1^\eta, \rho_s,)$ if $x \in X$
- $[\![f(\overline{u})]\!]^{\eta,\sigma}_{\rho_s} = \mathcal{A}_f([\![\overline{u}]\!]^{\eta,\sigma}_{\rho_s})$ if $f \in \mathcal{F}$

Such an interpretation of terms of course depends on the functional model $\mathcal{M}_f$: we may add $\mathcal{M}_f$ as an index of the semantic bracket if needed.

On the contrary, if the parameters are clear from the context we may simply write $[\![]\!]$ for $[\![]\!]^{\sigma}_{\rho_s}$, or provide with the relevant arguments only.

Example A.2. Let us consider $\mathcal{N} = \{sk, m, r\}$ and $\mathcal{F} = \{enc\}$. We may define $\mathcal{A}_{enc}$ as a TM implementing some encryption function, and $\mathcal{A}_{sk}$ as the TM which extracts the η first bits of ρ_s, and similarly for m and r with the following bits of ρ_s. In this cryptographic library, the term $enc(m, r, sk)$ will now be interpreted as the encryption of a random string by a random string.

B Protocols

B.1 Protocol Algebra

The precise syntax of our process algebra is defined in Figure 10.

B.2 Formal definition of a protocol execution

A protocol state is a pair $\varphi, (P_1, \sigma_1) \| \cdots \| (P_n, \sigma_n)$, where each P_i is a protocol, σ_i is an environment binding variables to bit-strings (intuitively the attacker's inputs), φ is a sequence of bit-strings (intuitively the protocol outputs). The parallel operator is considered as associative and commutative.

The semantics of elementary protocols assumes that, after an attacker input, the protocol moves immediately as much as possible until it stops or waits for another input. Formally, we define a relation $\rightarrow$, that does not depend on the attacker, such that, for instance, $\phi, (\mathtt{out}(c, t).P, \sigma) \rightarrow \phi \uplus (c, [\![t]\!]^{\eta,\sigma}_{\rho_s}), P, \sigma$. $\sigma \uplus \nu$ is defined such that all previously defined

terms:			
t	::=	n	names
	|	$n_{\vec{i}}$	indexed names
	|	x	variable
	|	$f(t_1, \ldots, t_n)$	operation of arity n
elementary protocols:			
P_{el}	::=	P_a	an atomic protocol
	|	$\mathsf{let}\ x = t\ \mathsf{in}\ P_{el}$	variable binding
	|	$\mathtt{in}(c, x).P_{el}$	input
	|	$\mathtt{out}(c, m).P_{el}$	output
	|	$\mathsf{if}\ s = t\ \mathsf{then}\ P_{el}\ \mathsf{else}\ P_{el}$	conditionals
	|	$\mathbf{0}$	
	|	$\bot$	
protocols:			
P, P'	::=	P_{el}	
	|	$P_{el}; P$	sequential composition
	|	$P \| P'$	parallel composition
	|	$\|^{i \leq N} P$	parallel replication
	|	$\|^{i} P$	unbounded replication
	|	$;^{i \leq N} P_{el}$	sequential replication
	|	$;^{i} P_{el}$	unbounded sequential replication

Figure 10: Protocol algebra

bindings in σ are overwritten by the ones in ν. In other words, in case of an output, we add to the frame the interpretation of t, given the current assignment of its variables and a (secret) random tape. We write $\xrightarrow{!}$ the reduction of a global state to its normal form w.r.t. $\rightarrow$.

Given an adversary $\mathcal{A}$, a sampling ρ_s of the names and a sampling ρ_r the attacker's random coins, for composed protocols, the operational semantics is given in Figure 11 and Figure 12 in the SOS style.

Note that a protocol with free variables may not be executed alone, but only in a context where its variables have been defined. Given P a protocol with free variables $x_1, \ldots, x_k$ and $n_1, \ldots, n_k$ a sequence of names, we may write $P(n_1, \ldots, n_k)$ as a short cut for $\mathsf{let}\ x_1 = n_1\ \mathsf{in}\ \ldots \mathsf{let}\ x_k = n_k\ \mathsf{in}\ P(x_1, \ldots, x_k)$.

Definition 33 (Context)**.** A context $C[\__1, \ldots, \__n]$ is a protocol built over the protocol algebra, where some elementary protocols are replaced with holes. Each hole $\__i$ can occur only once in the context. Given the (elementary) protocols $P_1, \ldots, P_n$, $C[P_1, \ldots, P_n]$ is then the protocol obtained when replacing each hole by the corresponding protocol.

B.3 Formal definition of protocol oracles

Definition 34 (Protocol Oracle)**.** A *protocol oracle* is defined as the previous stateless oracles, except that it has an additional *history* input, and only use $\overline{w}$ from its input $(\overline{w}, r, s)$. The *protocol oracle machines* also have an additional *history tape*, that cannot be accessed by the machine: it is only passed to the oracle, which also records the input queries on the history

Elementary protocols

$$\frac{}{\varphi, (\mathsf{if}\ s = t\ \mathsf{then}\ P\ \mathsf{else}\ Q, \sigma)\ \to\ \varphi, P, \sigma}\ \text{if } [\![s]\!]^{\eta,\sigma}_{\rho_s} = [\![t]\!]^{\eta,\sigma}_{\rho_s}$$

$$\frac{}{\varphi, (\mathsf{if}\ s = t\ \mathsf{then}\ P\ \mathsf{else}\ Q, \sigma)\ \to\ \varphi, Q, \sigma}\ \text{if } [\![s]\!]^{\eta,\sigma}_{\rho_s} \neq [\![t]\!]^{\eta,\sigma}_{\rho_s}$$

$$\frac{\varphi, P, \sigma \uplus \{x \mapsto \mathcal{A}(\rho_r, \varphi)\}\ \xrightarrow{!}\ \varphi', P', \sigma'}{\varphi,\ \mathsf{in}\ (c, x).P, \sigma\ \underset{\mathcal{A}}{\longrightarrow}\ \varphi', P', \sigma'}$$

$$\frac{}{\varphi, \mathtt{out}(c, s).P, \sigma\ \to\ \varphi \uplus \{[\![s]\!]^{\eta,\sigma}_{\rho_s}\}, P, \sigma}$$

$$\frac{}{\varphi, \mathsf{let}\ x = t\ \mathsf{in}\ P, \sigma\ \to\ \varphi, P, \sigma \uplus \{x \mapsto [\![t]\!]^{\eta,\sigma}_{\rho_s}\}}$$

$$\frac{\varphi, P, \sigma\ \xrightarrow{!}\ \varphi', P', \sigma'}{\varphi, P, \sigma\ \underset{\mathcal{A}}{\longrightarrow}\ \varphi', P', \sigma'}$$

Sequential compositions

$$\frac{\varphi, P, \sigma\ \underset{\mathcal{A}}{\longrightarrow}\ \varphi', P', \sigma'}{\varphi, P; Q, \sigma\ \underset{\mathcal{A}}{\longrightarrow}\ \varphi', P'; Q, \sigma'}$$

$$\frac{}{\varphi, \mathbf{0}; Q, \sigma\ \underset{\mathcal{A}}{\longrightarrow}\ \varphi, Q, \sigma}$$ Variable bindings are passed to Q when the prefix execution succeeds

$$\frac{}{\varphi, \bot; Q, \sigma\ \underset{\mathcal{A}}{\longrightarrow}\ \varphi, \bot, \sigma}$$ Q cannot be executed when the prefix execution fails

$$\frac{}{\varphi, ;^{i \leq N} P, \sigma\ \underset{\mathcal{A}}{\longrightarrow}\ \varphi, P\{i \mapsto 1\}; \cdots ; P\{i \mapsto N\}, \sigma}$$

$$\frac{}{\varphi, ;^{i} P, \sigma\ \underset{\mathcal{A}}{\longrightarrow}\ \varphi, ;^{i \leq \mathcal{A}(\rho_r, \varphi)} P, \sigma}$$

Figure 11: Operational semantics of elementary protocols and sequential compositions

Protocols

$$\frac{}{\varphi, (\mathbf{0}, \sigma) \| E \underset{\mathcal{A}}{\rightarrow} \varphi, E}$$

Parallel processes are outside of the scope of local bindings

$$\frac{}{\varphi, (\bot, \sigma) \| E \underset{\mathcal{A}}{\rightarrow} \varphi, E}$$

$$\frac{\varphi, P, \sigma \underset{\mathcal{A}}{\rightarrow} \varphi', P', \sigma'}{\varphi, (P, \sigma) \| E \underset{\mathcal{A}}{\rightarrow} \varphi', (P', \sigma') \| E}$$

The interactions between a process P and processes running in parallel are computed by the attacker

$$\frac{}{\varphi, P \| Q, \sigma \underset{\mathcal{A}}{\rightarrow} \varphi, (P, \sigma) \| (Q, \sigma)}$$

$$\frac{}{\varphi, \|^{i \leq N} P, \sigma \underset{\mathcal{A}}{\rightarrow} \varphi, P\{i \mapsto 1\} \| \cdots \| P\{i \mapsto N\}, \sigma}$$

$$\frac{}{\varphi, (\|^{i} P) \| E, \sigma \underset{\mathcal{A}}{\rightarrow} \varphi, (\|^{i \leq \mathcal{A}(\rho_r, \varphi)} P) \| E, \sigma}$$

Figure 12: Operational Semantics of protocols

tape. We write $\mathcal{A}^{\mathcal{O}_P(\rho_s)}$ for a protocol oracle Turing machine whose initial history tape is empty and such that $\mathcal{O}_P$ does not use the random tape $\rho_{\mathcal{O}}$.

We may generalize in the natural way the definition of *protocol oracle machines* to support any number of oracles where each *protocol oracle* has a distinct additional history tape.

We are now ready, given a protocol P, to define the protocol oracle $\mathcal{O}_P$.

Definition 35. Given a protocol P (which is action deterministic), a functional model $\mathcal{M}_f$, a security parameter $\eta \in \mathbb{N}$ and a random tape ρ_s, O_P is the protocol oracle, which, given ρ_s and an history $\theta = \{o_1, ..., o_n\} \in (\{0,1\}^*)^n$, on a query m:

- appends m to the history tape;
- executes the protocol P according to the semantics, using as inputs the history;
- return the final output produced by the protocol.

We extend the definition of PPTOM with:

- A protocol oracle input tape
- A protocol oracle history tape
- A protocol oracle output tape

The machine may call the protocol oracle $\mathcal{O}_P$ by writing on its input tape some content m, and there is then a single move to the current configuration to a configuration in which the history tape has been extended with the content of the input tape, and the protocol oracle output tape has been set to the output of $\mathcal{O}_P(\rho_s, \theta)(m)$.

We will often need to consider that we may have several protocol oracles for one PPTOM. We thus define a way to compose together oracles and protocol oracles. Protocol oracles can be merged together only if their respective protocols do not share input channels.

Definition 36. For any n and protocols $P_1, \ldots, P_n$ such that $\forall 1 \leq i < j \leq n. \mathcal{C}(P_i) \cap \mathcal{C}(P_j) = \emptyset$, we define the oracle $< \mathcal{O}_{P_1}, \ldots, \mathcal{O}_{P_n} > (\rho_s, \theta)$ which on input *query*:

- check if its input is of the form $query := (channel, mess)$;
- computes i such that $channel \in C(P_i)$, and reject if there is no such i;
- computes θ_i the projection of its history such that $\theta_i = \{(channel, mess) \in \theta | channel \in \mathcal{C}(P_i)\}$;
- return the value of $\mathcal{O}_{P_i}(\rho_s, \theta_i)(mess)$.

We will often write $\mathcal{A}^{\mathcal{O}_{P_1}, \ldots \mathcal{O}_{P_n}}(\omega, \rho_r)$ for $\mathcal{A}^{<\mathcal{O}_{P_1}, \ldots \mathcal{O}_{P_n}>}(\omega, \rho_r)$.

We may then use PPTOM with multiple oracles and multiple protocol oracles, written $\mathcal{A}^{\mathcal{O}_1, \ldots, \mathcal{O}_k, \mathcal{O}_{P_1}, \ldots \mathcal{O}_{P_n}}(\omega, \rho_r)$ for $\mathcal{A}^{<\mathcal{O}_1, \ldots \mathcal{O}_k>, <\mathcal{O}_{P_1}, \ldots \mathcal{O}_{P_n}>}(\omega, \rho_r)$.

C A case study : signed DDH

We apply our framework to the ISO 9798-3 protocol. It was proven UC composable in [13]. With our framework, it could be composed even with an oracle which uses the same long term secret. We also note that our proof could be mechanized, as it is performed in a first order logic. With only one session and if we prioritise the outputs, there are three interleaving, with corresponding frames ϕ_3, ψ_3 and χ_3:

$$
\begin{aligned}
\phi_0 &:= \mathtt{pk}(sk_I), g^a \\
\phi_1 &:= \phi_0, (\mathtt{pk}(sk_R), g^b, \mathtt{sign}((g_0(\phi_0), g^b, g_1(\phi_0)), sk_R)) \\
\phi_2 &:= \phi_1, \mathsf{if}\ \mathtt{checksign}(g_4(\phi_1), g_2(\phi_1)) \wedge g_3(\phi_1) = \pi_2(\mathtt{getmess}(g_4(\phi_1)))\ \mathsf{then} \\
&\quad \mathsf{if}\ \pi_1(\mathtt{getmess}(g_4(\phi_1))) = g^a \wedge \pi_3(\mathtt{getmess}(g_4(\phi_1))) = \mathtt{pk}(sk_I)\ \mathsf{then} \\
&\quad\quad \mathtt{sign}((g_3(\phi_1), g^a, g_2(\phi_1)), sk_I), \_A \\
\phi_3 &:= \phi_2, \mathsf{if}\ \mathtt{checksign}(g_5(\phi_2), g_1(\phi_0) \wedge g_0(\phi_0) = \pi_2(\mathtt{getmess}(g_5(\phi_2)))\ \mathsf{then} \\
&\quad \mathsf{if}\ \pi_1(\mathtt{getmess}(g_5(\phi_2))) = g^b \wedge \pi_3(\mathtt{getmess}(g_5(\phi_2))) = \mathtt{pk}(sk_R)\ \mathsf{then} \\
&\quad\quad \_B.
\end{aligned}
$$

$$
\begin{aligned}
\psi_0 &:= \phi_0 \\
\psi_1 &:= \psi_0, \mathsf{if}\ \mathtt{checksign}(g_2(\psi_0), g_0(\psi_0)) \wedge g_1(\psi_0) = \pi_2(\mathtt{getmess}(g_2(\psi_0)))\ \mathsf{then} \\
&\quad \mathsf{if}\ \pi_1(\mathtt{getmess}(g_2(\psi_0))) = g^a \wedge \pi_3(\mathtt{getmess}(g_2(\psi_0))) = \mathtt{pk}(sk_I)\ \mathsf{then} \\
&\quad\quad \mathtt{sign}((g_1(\psi_0), g^a, g_0(\psi_0)), sk_I), \_A \\
\psi_2 &:= \psi_1, (\mathtt{pk}(sk_R), g^b, \mathtt{sign}((g_3(\psi_1), g^b, g_4(\psi_1)), sk_R)) \\
\psi_3 &:= \mathsf{if}\ \mathtt{checksign}(g_5(\psi_2), g_4(\psi_1) \wedge g_3(\psi_1) = \pi_2(\mathtt{getmess}(g_5(\psi_2)))\ \mathsf{then} \\
&\quad \mathsf{if}\ \pi_1(\mathtt{getmess}(g_5(\psi_2))) = g^b \wedge \pi_3(\mathtt{getmess}(g_5(\psi_2))) = \mathtt{pk}(sk_R)\ \mathsf{then} \\
&\quad\quad \_B.
\end{aligned}
$$

$$
\begin{aligned}
\chi_0 &:= \phi_0 \\
\chi_1 &:= \chi_0, (\mathtt{pk}(sk_R), g^b, \mathtt{sign}((g_0(\chi_0), g^b, g_1(\chi_0)), sk_R)) \\
\chi_2 &:= \mathsf{if}\ \mathtt{checksign}(g_2(\chi_1), g_1(\chi_0) \wedge g_0(\chi_0) = \pi_2(\mathtt{getmess}(g_2(\chi_1)))\ \mathsf{then} \\
&\quad \mathsf{if}\ \pi_1(\mathtt{getmess}(g_2(\chi_1))) = g^b \wedge \pi_3(\mathtt{getmess}(g_2(\chi_1))) = \mathtt{pk}(sk_R)\ \mathsf{then} \\
&\quad\quad \_B. \\
\chi_3 &:= \chi_2, \mathsf{if}\ \mathtt{checksign}(g_5(\chi_2), g_3(\chi_2)) \wedge g_4(\chi_2) = \pi_2(\mathtt{getmess}(g_5(\chi_2)))\ \mathsf{then} \\
&\quad \mathsf{if}\ \pi_1(\mathtt{getmess}(g_5(\chi_2))) = g^a \wedge \pi_3(\mathtt{getmess}(g_5(\chi_2))) = \mathtt{pk}(sk_I)\ \mathsf{then} \\
&\quad\quad \mathtt{sign}((g_4(\chi_2), g^a, g_3(\chi_2)), sk_I), \_A
\end{aligned}
$$

C.1 Key exchange security

We show how to apply Corollary 1. We will use $id^I = sk_I$, $id^R = sk_R$, $lsid^I = g^a$ and $lsid^R = g^b$. Then, for any n we set $\overline{s} = a_1, b_1, \ldots, a_n, b_n$. We define the functions:

$$
\begin{aligned}
T^I(m, \overline{s}) :=&\quad \exists a_i \in \overline{s}, \exists A, A' m = (A, g^{a_i}, A') \\
T^R(m, \overline{s}) :=&\quad \exists b_i \in \overline{s}, \exists A, A' m = (A, g^{b_i}, A')
\end{aligned}
$$

We then set $\mathcal{O}_{KE} = \mathcal{O}^{\mathsf{sign}}_{T^I, sk_I, \overline{s}}, \mathcal{O}^{\mathsf{sign}}_{T^R, sk_R, \overline{s}}, \mathcal{O}_{\overline{s}}$, where $\mathcal{O}_{\overline{s}}$ simply reveals the exponents of the elements in $\overline{s}$.

We fix $KE(sk_I, sk_R, a_i, b_i)[\__I, \__R] = I(a_i, sk_I)[\__I] \| I(b_i, sk_R)[\__R]$.

We remark that proving at the end of the protocol that $(k_I, g^a, olsid, oid)$ is indistinguishable to $(k, g^a, olsid, oid)$ is equivalent to proving that k_I is indistinguishable from k, as the other elements are public information.

To apply the Corollary, it remains to prove that:

1. $\forall 1 \leq i \leq N, (\nu a_i, id^I, b_i, id^R. KE(sk_I, sk_R, a_i, b_i)[out(k_I, g^{a_i}, olsid, oid), out(k_R, g^{b_i}, olsid, oid)]$ is $\mathcal{O}_{KE}$ simulatable)).
2. Ax is $\mathcal{O}_{KE}$ sound.

3.

$$
\begin{aligned}
&Ax \models \phi_3[g_3(\phi_1)^a, g_0(\phi_0)^b] \sim \\
&\phi_3[\ \ \mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then} \\
&\qquad out(k) \\
&\quad \mathsf{else\ if}\ \neg \textstyle\bigvee_i g_3(\phi_1) = g^{b_i} \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then} \\
&\qquad bad \\
&\quad \mathsf{else}\ out(g_3(\phi_1)^a) \\
&, \\
&\quad \mathsf{if}\ g_0(\phi_0) = g^a \wedge g_1(\phi_0) = \mathtt{pk}(sk_I)\ \mathsf{then} \\
&\qquad out(k) \\
&\quad \mathsf{else\ if}\ \neg \textstyle\bigvee_i g_0(\phi_0) = g^{a_i} \wedge g_1(\phi_0) = \mathtt{pk}(sk_I)\ \mathsf{then} \\
&\qquad bad \\
&\quad \mathsf{else}\ out(g_0(\phi_0)^b)]
\end{aligned}
$$

4.

$$
\begin{aligned}
&Ax \models \psi_3[g_1(\psi_0)^a, g_3(\psi_1)^b] \sim \\
&\psi_3[\ \ \mathsf{if}\ g_1(\psi_0) = g^b \wedge g_0(\psi_0) = \mathtt{pk}(sk_R)\ \mathsf{then} \\
&\qquad out(k) \\
&\quad \mathsf{else\ if}\ \neg \textstyle\bigvee_i g_1(\psi_0) = g^{b_i} \wedge g_0(\psi_0) = \mathtt{pk}(sk_R)\ \mathsf{then} \\
&\qquad bad \\
&\quad \mathsf{else}\ out(g_1(\psi_0)^a) \\
&, \\
&\quad \mathsf{if}\ g_3(\psi_1) = g^a \wedge g_4(\psi_1) = \mathtt{pk}(sk_I)\ \mathsf{then} \\
&\qquad out(k) \\
&\quad \mathsf{else\ if}\ \neg \textstyle\bigvee_i g_3(\psi_1) = g^{a_i} \wedge g_4(\psi_1) = \mathtt{pk}(sk_I)\ \mathsf{then} \\
&\qquad bad \\
&\quad \mathsf{else}\ out(g_3(\psi_1)^b)]
\end{aligned}
$$

5.

$$
\begin{aligned}
&Ax \models \chi_3[g_4(\chi_2)^a, g_0(\chi_0)^b] \sim \\
&\chi_3[\ \ \mathsf{if}\ g_4(\chi_2) = g^b \wedge g_3(\chi_2) = \mathtt{pk}(sk_R)\ \mathsf{then} \\
&\qquad out(k) \\
&\quad \mathsf{else\ if}\ \neg \textstyle\bigvee_i g_4(\chi_2) = g^{b_i} \wedge g_3(\chi_2) = \mathtt{pk}(sk_R)\ \mathsf{then} \\
&\qquad bad \\
&\quad \mathsf{else}\ out(g_4(\chi_2)^a) \\
&, \\
&\quad \mathsf{if}\ g_0(\chi_0) = g^a \wedge g_1(\chi_0) = \mathtt{pk}(sk_I)\ \mathsf{then} \\
&\qquad out(k) \\
&\quad \mathsf{else\ if}\ \neg \textstyle\bigvee_i g_0(\chi_0) = g^{a_i} \wedge g_1(\chi_0) = \mathtt{pk}(sk_I)\ \mathsf{then} \\
&\qquad bad \\
&\quad \mathsf{else}\ out(g_0(\chi_0)^b)]
\end{aligned}
$$

We have that $Ax = \text{EUF-CMA}_{T^I,F,sk_I,\overline{s}} \wedge \text{EUF-CMA}_{T^R,F,sk_R,\overline{s}} \wedge DDH_{\overline{s},sk_I,sk_R}$ is $\mathcal{O}_{KE}$-sound thanks to Proposition 27.

The simulatability also instantly follows from the definitions, as the attackers as access to the a_i and b_i, and can produce signatures on them (but only on them).

We only provide the proof for the most difficult frame ϕ_3, where the attacker has the most knowledge for each computation. ψ_3 and χ_3 can be handled exactly the same way, except that on applications of the EUF-CMA axioms, the attacker does not have the honest signatures in the frame (as the order of the agents has been mixed up), which simplify the proof.

C.2 Proof for ϕ_3

We from now on omit Ax.

C.2.1 Real or random of the key

The real or random goal is:

$$\begin{array}{l}\phi_3[\ \ \mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \quad\ g_3(\phi_1)^a\\ \quad\mathsf{else}\\ \quad\ g_3(\phi_1)^a\\ ,\\ \mathsf{if}\ g_0(\phi_0) = g^a \wedge g_1(\phi_0) = \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad\ g_0(\phi_0)^b\\ \quad\mathsf{else}\\ \quad\ g_0(\phi_0)^b\\ ]\end{array} \quad\sim\quad \begin{array}{l}\phi_3[\ \ \mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \quad\ k\\ \quad\mathsf{else}\\ \quad\ g_3(\phi_1)^a\\ ,\\ \mathsf{if}\ g_0(\phi_0) = g^a \wedge g_1(\phi_0) = \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad\ k\\ \quad\mathsf{else}\\ \quad\ g_0(\phi_0)^b\\ ]\end{array}$$

There are by case disjunctions four possible cases, the first one being:

$$\begin{array}{l}\mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \mathsf{if}\ g_0(\phi_0) = g^a \wedge g_1(\phi_0) = \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad\phi_3[g_3(\phi_1)^a, g_0(\phi_0)^b]\end{array} \quad\sim\quad \begin{array}{l}\mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \mathsf{if}\ g_0(\phi_0) = g^a \wedge g_1(\phi_0) = \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad\phi_3[k, k]\end{array}$$

We can define the substitution $\tau := \{g_3(\phi_1) \leftarrow g^b, g_2(\phi_1), \leftarrow \mathtt{pk}(sk_R), g_0(\phi_0) \leftarrow g^a, g_1(\phi_0) \leftarrow \mathtt{pk}(sk_I)\}$, the goal then becomes: $\phi_3[g^{ab}, g^{ab}]\tau \sim \phi_3[k, k]\tau$.

Note that a, b is not included in $\overline{s}, sk_I, sk_R$, we can thus use the DDH axiom on them. Looking at $\phi_3\tau$, we also see that all occurences of a and b are of the form g^a or g^b. Thus applying DDH directly gives us:

$$\phi_3[g^{ab}, g^{ab}]\tau \sim \phi_3[g^c, g^c]\tau$$

We conclude by renaming of g^c into k.

The second case is:

$$\begin{array}{l}\mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \mathsf{if}\ g_0(\phi_0) \neq g^a \vee g_1(\phi_0) \neq \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad\phi_3[g_3(\phi_1)^a, g_0(\phi_0)^b]\end{array} \quad\sim\quad \begin{array}{l}\mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \mathsf{if}\ g_0(\phi_0) \neq g^a \vee g_1(\phi_0) \neq \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad\phi_3[k, g_0(\phi_0)^b]\end{array}$$

Here, we actually prove that we never go into the branch which reveals either the $g_3(\phi_1)^a$ or the k, thus yielding the equivalence.

We thus prove that:

$$\begin{array}{l}\mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \mathsf{if}\ g_0(\phi_0) \neq g^a \vee g_1(\phi_0) \neq \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad\mathtt{checksign}(g_4(\phi_1), g_2(\phi_1))\\ \quad\wedge g_3(\phi_1) = \pi_2(\mathtt{getmess}(g_4(\phi_1)))\\ \quad\wedge \pi_1(\mathtt{getmess}(g_4(\phi_1))) = g^a\\ \quad\wedge \pi_3(\mathtt{getmess}(g_4(\phi_1))) = \mathtt{pk}(sk_I)\end{array} \quad\sim\quad \begin{array}{l}\mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \mathsf{if}\ g_0(\phi_0) \neq g^a \vee g_1(\phi_0) \neq \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad\mathbf{false}\end{array}$$

We have by application of the equalities:

$$\begin{array}{l}\mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \mathsf{if}\ g_0(\phi_0) \neq g^a \vee g_1(\phi_0) \neq \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad \mathtt{checksign}(g_4(\phi_1), g_2(\phi_1))\\ \quad \wedge g_3(\phi_1) = \pi_2(\mathtt{getmess}(g_4(\phi_1)))\\ \quad \wedge \pi_1(\mathtt{getmess}(g_4(\phi_1))) = g^a\\ \quad \wedge \pi_3(\mathtt{getmess}(g_4(\phi_1))) = \mathtt{pk}(sk_I)\end{array} \sim \begin{array}{l}\mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \mathsf{if}\ g_0(\phi_0) \neq g^a \vee g_1(\phi_0) \neq \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad \mathtt{checksign}(g_4(\phi_1), \mathtt{pk}(sk_R))\\ \quad \wedge g^b = \pi_2(\mathtt{getmess}(g_4(\phi_1)))\\ \quad \wedge \pi_1(\mathtt{getmess}(g_4(\phi_1))) = g^a\\ \quad \wedge \pi_3(\mathtt{getmess}(g_4(\phi_1))) = \mathtt{pk}(sk_I)\end{array}$$

We now apply EUF-CMA$_{T^R,F,sk_R,\overline{s}}$ to $g_4(\phi_1)$, so we either have $g_4(\phi_1) = \mathtt{sign}((g_0(\phi_0), g^b, g_1(\phi_0)), sk_R)$ (the honest signature), which is a contradiction with $g_0(\phi_0) \neq g^a$ and $\pi_1(\mathtt{getmess}(g_4(\phi_1))) = g^a$, or the signature comes from the oracle and $g_4(\phi_1) = (A, g^{b_i}, A')$, in contradiction with $g^b = \pi_2(\mathtt{getmess}(g_4(\phi_1)))$.

We thus have:

$$\begin{array}{l}\mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \mathsf{if}\ g_0(\phi_0) \neq g^a \vee g_1(\phi_0) \neq \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad \mathtt{checksign}(g_4(\phi_1), \mathtt{pk}(sk_R))\\ \quad \wedge g^b = \pi_2(\mathtt{getmess}(g_4(\phi_1)))\\ \quad \wedge \pi_1(\mathtt{getmess}(g_4(\phi_1))) = g^a\\ \quad \wedge \pi_3(\mathtt{getmess}(g_4(\phi_1))) = \mathtt{pk}(sk_I)\end{array} \sim \begin{array}{l}\mathsf{if}\ g_3(\phi_1) = g^b \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \mathsf{if}\ g_0(\phi_0) \neq g^a \vee g_1(\phi_0) \neq \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad \mathbf{false}\end{array}$$

And we conclude by transitivity.

Of the two remaining cases, one is symmetrical to the previous one, and the last one is trivial.

C.2.2 Authentication

The goal is:

$$\begin{array}{l}\phi_3[\ \mathsf{if}\ g_3(\phi_1) \neq g^b\ \mathsf{then}\\ \quad \mathsf{if}\ \neg \bigvee_i g_3(\phi_1) = g^{b_i} \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \quad\quad g_3(\phi_1)^a\\ \quad ,\\ \quad \mathsf{if}\ g_0(\phi_0) \neq g^a\ \mathsf{then}\\ \quad \mathsf{if}\ \neg \bigvee_i g_0(\phi_0) = g^{a_i} \wedge g_1(\phi_0) = \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad\quad g_0(\phi_0)^b\\ ]\end{array} \sim \begin{array}{l}\phi_3[\ \mathsf{if}\ \neg g_3(\phi_1) = g^b\ \mathsf{then}\\ \quad \mathsf{if}\ \neg \bigvee_i g_3(\phi_1) = g^{b_i} \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\ \mathsf{then}\\ \quad\quad bad\\ \quad ,\\ \quad \mathsf{if}\ g_0(\phi_0) \neq g^a\ \mathsf{then}\\ \quad \mathsf{if}\ \neg \bigvee_i g_0(\phi_0) = g^{a_i} \wedge g_1(\phi_0) = \mathtt{pk}(sk_I)\ \mathsf{then}\\ \quad\quad bad\\ ]\end{array}$$

We prove that each condition is never true using the EUF-CMA axioms. The four cases are symmetrical, we only prove the first one:

$$\begin{array}{l}\mathsf{if}\ \mathtt{checksign}(g_4(\phi_1), g_2(\phi_1)) \wedge g_3(\phi_1) = \pi_2(\mathtt{getmess}(g_4(\phi_1)))\ \mathsf{then}\\ \mathsf{if}\ \pi_1(\mathtt{getmess}(g_4(\phi_1))) = g^a \wedge\\ \pi_3(\mathtt{getmess}(g_4(\phi_1))) = \mathtt{pk}(sk_I)\ \mathsf{then}\\ \mathsf{if}\ g_3(\phi_1) \neq g^b\ \mathsf{then}\\ \neg \bigvee_i g_3(\phi_1) = g^{b_i} \wedge g_2(\phi_1) = \mathtt{pk}(sk_R)\end{array}$$

$$\sim$$

$$\begin{array}{l}\mathsf{if}\ \mathtt{checksign}(g_4(\phi_1), g_2(\phi_1)) \wedge g_3(\phi_1) = \pi_2(\mathtt{getmess}(g_4(\phi_1)))\ \mathsf{then}\\ \mathsf{if}\ \pi_1(\mathtt{getmess}(g_4(\phi_1))) = g^a \wedge\\ \pi_3(\mathtt{getmess}(g_4(\phi_1))) = \mathtt{pk}(sk_I)\ \mathsf{then}\\ \mathsf{if}\ g_3(\phi_1) \neq g^b\ \mathsf{then}\\ \mathbf{false}\end{array}$$

By case disjunction $g_2(\phi_1) = \mathtt{pk}(sk_R)$, the negative one being trivial, we must prove:

$$\begin{array}{l}
\mathsf{if}\ \mathtt{checksign}(g_4(\phi_1), \mathtt{pk}(sk_R)) \wedge g_3(\phi_1) = \pi_2(\mathtt{getmess}(g_4(\phi_1)))\ \mathsf{then} \\
\mathsf{if}\ \pi_1(\mathtt{getmess}(g_4(\phi_1))) = g^a \wedge \\
\pi_3(\mathtt{getmess}(g_4(\phi_1))) = \mathtt{pk}(sk_I)\ \mathsf{then} \\
\mathsf{if}\ g_3(\phi_1) \neq g^b\ \mathsf{then} \\
\neg \bigvee_i g_3(\phi_1) = g^{b_i} \\
\qquad\qquad\qquad\qquad \sim \\
\mathsf{if}\ \mathtt{checksign}(g_4(\phi_1), \mathtt{pk}(sk_R)) \wedge g_3(\phi_1) = \pi_2(\mathtt{getmess}(g_4(\phi_1)))\ \mathsf{then} \\
\mathsf{if}\ \pi_1(\mathtt{getmess}(g_4(\phi_1))) = g^a \wedge \\
\pi_3(\mathtt{getmess}(g_4(\phi_1))) = \mathtt{pk}(sk_I)\ \mathsf{then} \\
\mathsf{if}\ g_3(\phi_1) \neq g^b\ \mathsf{then} \\
\mathbf{false}
\end{array}$$

We now apply EUF-CMA$_{T^R,F,sk_R,\overline{s}}$ to $g_4(\phi_1)$, so we either have $g_4(\phi_1) = \mathtt{sign}((g_0(\phi_0), g^b, g_1(\phi_0)), sk_R)$ (the honest signature), which is a contradiction with $g_3(\phi_1) \neq g^b$ and $g_3(\phi_1) = \pi_2(\mathtt{getmess}(g_4(\phi_1)))$, or the signature comes from the oracle and $g_4(\phi_1) = (A, g^{b_i}, A')$, in contradiction with $\neg \bigvee_i g_3(\phi_1) = g^{b_i}$. This conclude the proof.

C.3 Conclusion for Signed DDH

We thus have the security of the signed DDH protocol. If we want to use Corollary 2 to compose it with for instance a record protocol $RP := RP_I(k) \| RP_R(k)$, which simply exchange encrypted messages using the exchanged key, and do not share any long term secret, it is trivial. Indeed, without any shared secret, $in(k); RP_I(k) \| in(k); RP_R(k)$ is simulatable without any oracle, so we can take $\mathcal{O}_p = \emptyset$. This means that we have the first set of hypothesis.

Now, RP would be proven secure with IND-CCA, and this can be proven easily, even if many other session of RP with distinct keys are in parallel. So we can simply set $\mathcal{O}_r$ as the oracle which outputs all the $k_{i,j}$ and $\mathcal{O}_k$ as the oracle which output $\overline{p}$, and obtain the multi session security of RP, and the simulatability of the key exchange.

RP could be a single round trip enrypted exchange, or actually any number of round trip, easily proved secure using Proposition 18.

D An application to SSH

D.1 Presentation of SSH

We only show here how the proof of security of SSH could be split up into smaller proofs thanks to our framework, but we do not actually prove the smaller proofs. We will thus only provide a high level point of view of SSH, not going into too many implementations details, but rather focusing on the parts that represent a challenge for composition. SSH is a simple key exchange which can be used to set up an authenticated and secret channel between a user's computer and a server, with first an authentication of the server, and then an optional authentication of the user, either through a password or a secret key. We provide in Figure 13 the basic SSH key exchange, with authentication through secret keys.

We can see that the indistinguishability of the key is not preserved through the protocol. The difficulty of SSH is moreover that once a user has established a secure connection to a server, he can from this server establish a secure connection to another server, while using the

$P(skP, pk(skS))[\_] :=$
  new a;
  $\mathtt{out}(g^a)$;
  $\mathtt{in}(< B, pk(skS), sign >)$
  let $k = B^a$ in
  let $sid = \mathtt{hash}(< g^a, B, k >)$ in
  if $\mathtt{checksign}(sign, pk(skS))$
    $\wedge\ \mathtt{getmess}(sign) = sid$ then
    $\mathtt{out}(\mathtt{enc}(\mathtt{sign}(sid, skP), k))$
    $[\_]$.

$S(skS, pk(skP))[\_] :=$
  $\mathtt{in}(A)$;
  new b;
  let $k = A^b$ in
  let $sid = \mathtt{hash}(< A, g^b, k >)$ in
  $\mathtt{out}(< g^b, pk(skS), \mathtt{sign}(sid, skS) >)$
  $\mathtt{in}(\mathtt{enc}(sign, k))$
  if $\mathtt{checksign}(sign, pk(skP))$
    $\wedge\ \mathtt{getmess}(sign) = sid$ then
    $[\_]$.

$$SSH :=!P(skP, pk(skS))[0] \| !S(skS, pk(skP))[0]$$

Figure 13: Basic SSH key exchange

secure channel previously established to obtain the user credentials. We provide in Figure 14 a model of the SSH with forwarding of agent (reusing the definitions of P and S from Figure 13), where after a P is ran successufully, a $ForwardAgent$ is started on the computer which can receive on the secret channel a signing request and perform the signature of it. In parallel, after the completion of some S, a distant session of P can be initiated by $PDistant$, which will request on the previous secret channel the signature of the sid. Finally, as the forwarding can be chained multiple time, at the end of a successful $PDistant$, a $ForwardServer$ is set up, which will accept to receive a signing request on the new secret channel of $PDistant$, forward the request on the old secret channel, get the signature and forward it.

With the agent forwarding, we are faced with the new problem which is that we sequentially compose a basic SSH exchange with other ones which use the same long term secret keys.

To summarize, to be able to prove the security of SSH with agent forwarding, we must be able to handle key confirmations and composition with shared long term secret.

D.2 The security of the protocol without forwarding agent

We show how we may apply Corollary 3 to the basic SSH protocol.

We provide in Figure 15, how we decompose the ssh protocols in order to prove its security. To simplify, we directly specify that P and S may only relate to each other by hard-coding the expected public keys inside them.

A first step is to obtain the hypothesis A-3, relating to the security of the basic SSH key exchange. We split this goal into two subgoals with a case study, the first one capturing the real or random of the key,

$$P^0(a, skP, pk(skS)); out(k) \| S^0(b, skS, pk(skP)); out(k) \cong_{\mathcal{O}}$$
$$P^0(a, skP, pk(skS)); [\mathsf{if}\ B = g^b\ \mathsf{then}\ out(k')\ \mathsf{else}\ out(k)]$$
$$\| S^0(b, skS, pk(skP))[\mathsf{if}\ A = g^a\ \mathsf{then}\ out(k')\ \mathsf{else}\ out(k)]$$

$PDistant(oldk, pk(skS)) :=$
  new a;
  $\mathtt{out}(g^a)$;
  $\mathtt{in}(B, pk(skS), sign)$
  let $k = B^a$ in
  let $sid = \mathtt{hash}(< g^a, B, kP >)$ in
  if $\mathtt{checksign}(sign, pk(skS))$
    $\wedge\ \mathtt{getmess}(sign) = sid$ then
    $\mathtt{out}(\mathtt{enc}(sid, oldk))$
    $\mathtt{in}(\mathtt{enc}(sign, oldk))$
    $\mathtt{out}(\mathtt{enc}(sign, k))$
    0.

$SForward(skS, pk(skP)) :=$
  $\mathtt{in}(A)$;
  new b;
  let $k = A^b$ in
  let $sid = \mathtt{hash}(< A, g^b, k >)$ in
  $\mathtt{out}(< g^b, pk(skS), \mathtt{sign}(sid, skS) >)$
  $\mathtt{in}(\mathtt{enc}(sign, k))$
  if $\mathtt{checksign}(sign, pk(skP))$
    $\wedge\ \mathtt{getmess}(sign) =< sid, \text{“}forwarded\text{”} >$ then
    0.

$ForwardAgent(skP, k) :=$
  $\mathtt{in}(\mathtt{enc}(sid, k))$
  $\mathtt{out}(\mathtt{enc}(\mathtt{sign}(< sid, \text{“}forwarded\text{”} >, skP), k))$

$SSH_{Forward} :=$
  $P(skP, pk(skS)); ForwardAgent(skP, k)$
  $\|SForward(skS, pk(skP))$
  $\|S(skS, pk(skP)); PDistant(k, pk(skS))$

Figure 14: SSH key exchange with agent forwarding

$P^0(a, skP, pk(skS)) :=$
  $\mathtt{out}(g^a)$;
  $\mathtt{in}(< B >)$
  let $k = B^a$ in
  0.
$P^1(a, skP, pk(skS), B, k) :=$
  $\mathtt{in}(< pk(skS), sign >)$
    let $sid = \mathtt{hash}(< g^a, B, k >)$ in
    if $\mathtt{checksign}(sign, pk(skS))$
    $\wedge\ \mathtt{getmess}(sign) = sid$ then
      $\mathtt{out}(\mathtt{enc}(\mathtt{sign}(sid, skP), k))$
      0.

$S^0(b, skS, pk(skP)) :=$
  $\mathtt{in}(A)$;
  let $k = A^b$ in
  let $sid = \mathtt{hash}(< A, g^b, k >)$ in
    $\mathtt{out}(g^b)$
$S^1(b, skS, pk(skP), sid, k) :=$
    $\mathtt{out}(< pk(skS), g^b, \mathtt{sign}(sid, skS) >)$
    $\mathtt{in}(\mathtt{enc}(sign, k))$
    if $\mathtt{checksign}(sign, pk(skP))$
    $\wedge\ \mathtt{getmess}(sign) = sid$ then
      0..

Figure 15: Divided SSH key exchange

and the second one the authentication:

$P^0(a, skP, pk(skS)); \mathsf{if}\ \neg(\bigvee_i B = g^{b_i})\ \mathsf{then}\ P^1(a, skP, pk(skS), B, k); out(k)$
$\|S^0(b, skS, pk(skP)); \mathsf{if}\ \neg(\bigvee_i A = g^{a_i})\ \mathsf{then}\ S^1(b, skS, pk(skP), sid, k); out(k)$
$\cong_{\mathcal{O}}$
$P^0(a, skP, pk(skS)); \mathsf{if}\ \neg(\bigvee_i B = g^{b_i})\ \mathsf{then}\ P^1(a, skP, pk(skS), B, k); bad$
$\|S^0(b, skS, pk(skP)); \mathsf{if}\ \neg(\bigvee_i A = g^{a_i})\ \mathsf{then}\ S^1(b, skS, pk(skP), sid, k); bad$

D.3 Proof of real of random

We start by proving that:

$$Ax \models P^0(a, skP, pk(skS)); out(k) \| S^0(b, skS, pk(skP)); out(k) \sim$$
$$P^0(a, skP, pk(skS)); [\mathsf{if}\ B = g^b\ \mathsf{then}\ out(k')\ \mathsf{else\ if}\ B = g^{b_i}\ \mathsf{then}\ out(k)]$$
$$\| S^0(b, skS, pk(skP))[\mathsf{if}\ A = g^a\ \mathsf{then}\ out(k')\ \mathsf{else\ if}\ A = g^{a_i}\ \mathsf{then}\ out(k)]$$

For this proof, we may use $Ax = DDH_{\overline{s}}$, where $\overline{s}$ does not contain a and b.

We denote ϕ_i^j the i-eme term of the j-eme folding in the left game, and ψ_i^j for the right game. After splitting over each possible folding of actions, we have the sequence of terms:

- $\phi_0 = g^a$; $\phi_1 = \phi_0, g^b, g_0(\phi_0)^b$; $\phi_2 = \phi_1, g_1(\phi_1)^a$
- $\phi_1^1 = \phi_0, g_0(\phi_0)^a$, ; $\phi_2^1 = \phi_1^1, g^b, g_1(\phi_1^1)^b$
- $\psi_0 = \phi_0$; $\psi_1 = \psi_0, g^b, \mathsf{if}\ g_0(\phi_0) = g^a\ \mathsf{then}\ k'\ \mathsf{else\ if}\ g_0(\phi_0) = g^{a_i}\ \mathsf{then}\ g_0(\phi_0)^b$; $\psi_2 = \psi_1, \mathsf{if}\ g_1(\psi_1) = g^b\ \mathsf{then}\ k'\ \mathsf{else\ if}\ g_1(\psi_1) = g^{b_i}\ \mathsf{then}\ g_1(\psi_1)^a$
- $\psi_1^1 = \psi_0, \mathsf{if}\ g_0(\phi_0) = g^b\ \mathsf{then}\ k'\ \mathsf{else\ if}\ g_0(\phi_0) = g^{b_i}\ \mathsf{then}\ g_0(\psi_0)^a$; $\psi_2^1 = \psi_1^1, g^b, \mathsf{if}\ g_1(\psi_1^1) = g^a\ \mathsf{then}\ k'\ \mathsf{else\ if}\ g_1(\psi_1^1) = g^{a_i}\ \mathsf{then}\ g_1(\psi_1^1)^b$

And we have to prove that $Ax \models \phi_2 \sim \psi_2$ and $Ax \models \phi_2^1 \sim \psi_2^1$.

D.3.1 Proof of $Ax \models \phi_2 \sim \psi_2$

We apply the EQ that are true in the if branches, and we perform a case study on the first conditional of the sequence, yielding the four terms:

- $\psi_1' = \psi_0, g^b, \mathsf{EQ}(g_0(\phi_0), g^a), k'$
- $\psi_1'' = \psi_0, g^b, \mathsf{EQ}(g_0(\phi_0)), g^{a_i}), g^{a_i b}$
- $\phi_1' = \phi_0, g^b, \mathsf{EQ}(g_0(\phi_0), g^a), g^{ab}$
- $\phi_1'' = \phi_0, g^b, \mathsf{EQ}(g_0(\phi_0), g^{a_i}), g^{a_i b}$

With DDH , we can replace g^{ab} with k', and with transitivity, we have that

$$Ax \models \phi_1' \sim \psi_1'$$

.

Moreover, we trivially have

$$Ax \models \phi_1'' \sim \psi_1''$$

The we also apply the $\mathsf{EQ}$and perform another case study on the second conditional, yielding eight terms:

- $\psi_2' = \psi_1', \mathsf{EQ}(g_1(\psi_1'), g^b), k'$
- $\psi_2'' = \psi_1', \mathsf{EQ}(g_1(\psi_1'), g^{b_i}), g^{ab_i}$
- $\psi_2''' = \psi_1'', \mathsf{EQ}(g_1(\psi_1''), g^b), k'$
- $\psi_2'''' = \psi_1'', \mathsf{EQ}(g_1(\psi_1''), g^{b_i}), g^{ab_i}$
- $\phi_2' = \phi_1', \mathsf{EQ}(g_1(\phi_1'), g^b), g^{ab}$
- $\phi_2'' = \phi_1', \mathsf{EQ}(g_1(\phi_1'), g^{b_i}), g^{ab_i}$
- $\phi_2''' = \phi_1'', \mathsf{EQ}(g_1(\phi_1''), g^b), g^{ab}$
- $\phi_2'''' = \phi_1'', \mathsf{EQ}(g_1(\phi_1''), g^{b_i}), g^{ab_i}$

From now on, we omit Ax. We then prove the four equivalence required to conclude:

1. $\phi_2' \sim \psi_2'$
   We first use function application (FA) multiple times to get $\phi_2' \sim \psi_1', \mathsf{EQ}(g_1(\psi_1'), g^b), g^{ab}$. Then, we use DDH to replace g^{ab} with k' and transitivity to conclude that: $\phi_2' \sim \psi_2'$.

2. $\phi_2'' \sim \psi_2''$
   FA* on $\phi_1' \sim \psi_1'$ yields the conclusion $\phi_2'' \sim \psi_1', \mathsf{EQ}(g_1(\psi_1'), g^{b_i}), g^{ab_i}$.

3. $\phi_2''' \sim \psi_2'''$
   FA* on $\phi_1'' \sim \psi_1''$, yields $\phi_2''' \sim \psi_1'', \mathsf{EQ}(g_1(\psi_1''), g^b), g^{ab}$. After expressing the fact that $g^{a_i b} = (g^b)^{a_i}$ (i.e. all terms can be expressed as a context of g^a, g^b, g^{ab}), we use DDH to replace g^{ab} with k' and conclude.

4. $\phi_2'''' \sim \psi_2''''$
   FA* on $\phi_1'' \sim \psi_1''$ yields the conclusion $\phi_2'''' \sim \psi_1'', \mathsf{EQ}(g_1(\psi_1''), g^{b_i}), g^{ab_i}$.

We thus have $Ax \models \phi_2 \sim \psi_2$.

D.3.2 Proof of $Ax \models \phi_2^1 \sim \psi_2^1$

We first note that $\mathsf{EQ}(g_0(\phi_0), g^b) \sim \mathbf{false}$ as ϕ_0 does not contain b. Thus, the positive branch can be eliminated and we get $\psi_1^1 \sim \phi_1^1$. We then have $\psi_2^1 \sim \psi_2^{1'}$ where $\psi_2^{1'} = \phi_1^1, g^b, \mathsf{if}\ g_1(\phi_1^1) = g^a\ \mathsf{then}\ k'\ \mathsf{else}\ g_1(\phi_1^1)^b$.

We conclude once again with a case study, a DDH for one case, and trivial equality in the other case.

D.4 Proof for the authentication

We now prove that:

$$
\begin{array}{l}
Ax \models P^0(a, skP, pk(skS)); \mathsf{if}\ \neg(\bigvee_i B = g^{b_i})\ \mathsf{then}\ P^1(a, skP, pk(skS), B, k); out(k) \\
\|S^0(b, skS, pk(skP)); \mathsf{if}\ \neg(\bigvee_i A = g^{a_i})\ \mathsf{then}\ S^1(b, skS, pk(skP), sid, k); out(k) \\
\sim \\
P^0(a, skP, pk(skS)); \mathsf{if}\ \neg(\bigvee_i B = g^{b_i})\ \mathsf{then}\ P^1(a, skP, pk(skS), B, k); bad \\
\|S^0(b, skS, pk(skP)); \mathsf{if}\ \neg(\bigvee_i A = g^{a_i})\ \mathsf{then}\ S^1(b, skS, pk(skP), sid, k); bad
\end{array}
$$

The proof is very similar to the proof of authentication of the signed DH key exchange, we only outline the arguments here.

$$\begin{aligned} T_P(m,\overline{s}) := &\quad \exists i, \exists X, m = \mathsf{hash}(g^{a_i}, X, X^{a_i}) \\ T_S(m,\overline{s}) := &\quad \exists i, \exists X, m = \mathsf{hash}(X, g^{b_i}, X^{b_i}) \end{aligned}$$

We have that $Ax = \text{EUF-CMA}_{T_P,skP,\overline{s}} \wedge \text{EUF-CMA}_{T_S,skS,\overline{s}}$ is $\mathcal{O}^{\mathsf{sign}}_{T_P,sk_A,\overline{s}}, \mathcal{O}^{\mathsf{sign}}_{T_S,sk_B,\overline{s}}, \mathcal{O}_{a_i,b_i}$ sound thanks to Proposition 27.

We prove that bad may never occur, either in P or S. For bad to occur, the signature checks must succeed in one of the process, while the session identifier is not an honest one. In this case, we prove that the signature checks will always fail, i.e that , for $sign$, B and A terms produced by the attacker:

$$\neg(\bigvee_i B = g^{b_i}) \wedge \ \mathtt{checksign}(sign, pk(skS))) \wedge \ \mathtt{getmess}(sign) = \mathtt{hash}(< g^a, B, B^a >) \sim \mathbf{false}$$

or

$$\neg(\bigvee_i A = g^{a_i}) \wedge \ \mathtt{checksign}(sign, pk(skP)) \wedge \ \mathtt{getmess}(sign) = \mathtt{hash}(< A, g^b, A^b >) \sim \mathbf{false}$$

If those two equivalences are true for all possible values of the term $sign$ that can be taken depending on the traces, bad will never be raised. Let us for instance prove the first one. For all possible traces, the only honest signature by skS that might appear inside the message $sign$ is of the form $\mathsf{sign}(\mathtt{hash}(< A, g^b, A^b >), skS)$.

By using the $\text{EUF-CMA}_{T_S,skS,\overline{s}}$ axiom, we obtain

$$\begin{gathered} \neg(\textstyle\bigvee_i B = g^{b_i}) \wedge \ \mathtt{checksign}(sign, pk(skS))) \wedge \ \mathtt{getmess}(sign) = \mathtt{hash}(< g^a, B, B^a >) \\ \sim \\ \neg(\textstyle\bigvee_i B = g^{b_i}) \wedge (T_s(\ \mathtt{getmess}(sign)) \vee \ \mathtt{getmess}(sign) = \mathtt{hash}(< A, g^b, A^b >), skS) \\ \wedge \mathtt{getmess}(sign) = \mathtt{hash}(< g^a, B, B^a >) \end{gathered}$$

$(T_s(\ \mathtt{getmess}(sign))$ is directly in contradiction with $\neg(\bigvee_i B = g^{b_i})$, and the same goes for $\mathtt{getmess}(sign) = \mathtt{hash}(< A, g^b, A^b >), skS)$, we do obtain the expected conclusion.

E SSH with forwarding agent

E.1 Scheme of the proof

Here, we wish to compose SSH with another potential session of SSH using the forwarding agent. Then, a protocol which is secure if executed with a real or random key should be secure when using the key given by the SSH session using the forward agent.

We will write FA for $ForwardAgent$, SF for $SForward$, and PD for $PDistant$. We consider a record protocol, satisfying a property of the form $Y(k)\|Z(k) \cong Y'(k)\|Z'(k)$.

We also assume that the agents are only willing to communicate with the honnest identity, i.e $pk(skS)$ and $pk(skP)$ are predefined inside the processes. This is usually the case for SSH, where the user is asked to either validate or insert himself some public key.

Our goal is then:

$$!_{n^2} \quad \begin{array}{l} P(skP, pk(skS)); FA(skP, k); \\ \|S(skS, pk(skP)); PD(k, pk(skS)); Y(k_{PD}) \\ \|SF(skS, pk(skP)); Z(k_{SF}) \end{array} \cong !_{n^2} \quad \begin{array}{l} P(skP, pk(skS)); FA(skP, k); \\ \|S(skS, pk(skP)); PD(k, pk(skS)); Y'(k_{PD}) \\ \|SF(skS, pk(skP)); Z'(k_{SF}) \end{array}$$

We split the proof using two applications of Corollary 3.

The P and S will use randomness of the form a_i, b_i, and PD and SF randomness of the form a'_i and b'_i.

E.1.1 First application of Corollary 3

The application is performed with the following hypothesis, which allows to derive the desired conclusion.

A-3:

$$\begin{array}{ll} P^0(a, skP); & \mathsf{if}\ \neg T(B, \overline{s})\ \mathsf{then} \\ & \quad P^1(a, skP, B^a); out(B^a) \\ & \mathsf{else}\ out(B^a, g^a, B) \\ \|S^0(b, skS); & \mathsf{if}\ \neg T(A, \overline{s})\ \mathsf{then} \\ & \quad S^1(b, skS, A^b); out(A^b) \\ & \mathsf{else}\ out(A^b, g^b, A) \end{array} \cong_{\mathcal{O}_{PS}, \mathcal{O}_{forward}} \begin{array}{ll} P^0(a, skP); & \mathsf{if}\ B = g^b\ \mathsf{then} \\ & \quad out(k, g^a, B) \\ & \mathsf{else\ if}\ \neg T(B, \overline{s})\ \mathsf{then} \\ & \quad P^1(a, skP, B^a); bad \\ & \mathsf{else}\ out(A^b, g^b, A) \\ \|S^0(b, skS); & \mathsf{if}\ A = g^a\ \mathsf{then} \\ & \quad out(k, g^b, A) \\ & \mathsf{else\ if}\ \neg T(B, \overline{s})\ \mathsf{then} \\ & \quad S^1(b, skS, A^b); bad \\ & \mathsf{else}\ out(A^b, g^b, A) \end{array}$$

B-1:

$$!_{n^2} P^1(k); FA(k) \| S^1(k); PD(k); Y \| SF; Z \cong_{\mathcal{O}_{KE_1}} !_{n^2} P^1(k); FA(k) \| S^1(k); PD(k); Y' \| SF; Z'$$

With the oracles:

- $\mathcal{O}_{PS}$ allows to simulate (A-1) the other honnest sessions of P and S, it corresponds to $\mathcal{O}^{\mathsf{sign}}_{T_P, skS, \overline{s}}, \mathcal{O}^{\mathsf{sign}}_{T_S, skP, \overline{s}}, \mathcal{O}_{a_i, b_i}$ of Appendix D.4.
- $\mathcal{O}_{forward}$ allows to simulate (C-1) the continuation, i.e the protocols of the form $\mathtt{in}(k); P^1(k); FA(k) \| \mathtt{in}(k); S^1(k); PD(k); Y \| SF; Z$
- $\mathcal{O}_{KE_1}$ allows to simulate (C-2) $!_{n^2} P \| S$ (it is similar to $\mathcal{O}_{PS}$).

All simulations are performed under $\nu skS, skP$. To define $\mathcal{O}_{forward}$, we need to settle an issue. . Indeed, for hypothesis C-1, we need to provide an oracle that can simulates sessions of the forwarding protocols. However, in order to get the simulatability of $in(k).FA(skP, k)$, one must give a generic signing oracles to the attacker, which would obviously make the protocol unsecure. Based on the assumption that the forwarded sessions perform signatures tagged with "forwarded", as shown below, we however can provide a signing oracle only for such messages, allowing for the simulatability of the forwarding agent, and of the forwarded client and server. More specifically, recall the the forwarding agent is of the form:

$$\begin{array}{l} FA(skP, k) := \\ \quad \mathtt{in}(\mathtt{enc}(sid, k)) \\ \quad\quad \mathtt{out}(\mathtt{enc}(\mathtt{sign}(< sid, \text{“}forwarded\text{”} >, skP), k)) \end{array}$$

Then, we may obtain the simulatability with:

$$T_{for}(m, \overline{s}) := \quad \exists A, m =< m, \text{“}forwarded\text{”} >$$

Then, $\mathcal{O}_{forward}$ is simply $\mathcal{O}^{\mathsf{sign}}_{T_{for},skP,\overline{s}}, \mathcal{O}^{\mathsf{sign}}_{T_{for},skS,\overline{s}}, \mathcal{O}_{a'_i,b'_i}$

This difficulty actually stems from a well known weakness in the agent forwarding. When a user logs on a remote server, he set up on the server a socket which allows to ask for any signature. If another user has privileged access to the server, he may also use the socket, and obtain a signature for any session. In our model, we assume that only honest sessions of forwarder P can access an agent, which allows us to prove the security. Providing a proof of SSH without this modification still represent a challenge regarding composition.

Now, the proof of A-3 is instantly derived from the proof performed in Appendices D.3 and D.4, where we replace for instance EUF-CMA$_{T_P,skP,\overline{s}}$ with EUF-CMA$_{T_P \vee T'_P,skP,\overline{s}}$. The proofs will work as previously, based on the remark that $T'_P(\ \mathtt{getmess}(sign), \overline{s})$ is incompatible for instance with $\mathtt{getmess}(sign) = \mathtt{hash}(< g^a, B, B^a >)$.

The difficulty now lies in proving the security of what we do after the first SSH key exchange, i.e proving Hypothesis B-1. This is where we apply once again Corollary 3.

E.1.2 Second application of Corollary 3

We wish to prove that:

$$!_{n^2} P^1(k); FA(k) \| S^1(k); PD(k); Y \| SF; Z \cong_{\mathcal{O}_{KE_1}} !_{n^2} P^1(k); FA(k) \| S^1(k); PD(k); Y' \| SF; Z'$$

We use as hypothesis:
A-3:

$$\begin{array}{rl}
P^1(k); FA(k) \| S^1(k); PD^0(a, skP); & \mathsf{if}\ \neg T(B, \overline{s})\ \mathsf{then} \\
& \quad PD^1(a, skP, B^a); out(B^a) \\
& \mathsf{else}\ out(B^a, g^a, B) \\
\| SF^0(b, skS); & \mathsf{if}\ \neg T(A, \overline{s})\ \mathsf{then} \\
& \quad SF^1(b, skS, A^b); out(A^b) \\
& \mathsf{else}\ out(A^b, g^b, A) \\
\cong_{\mathcal{O}_{KE_1}, \mathcal{O}^k_{FPS}, \mathcal{O}_{YZ}} & \\
P^1(k); FA(k) \| S^1(k); PD^0(a, skP); & \mathsf{if}\ B = g^b\ \mathsf{then} \\
& \quad out(k, g^a, B) \\
& \mathsf{else\ if}\ \neg T(B, \overline{s})\ \mathsf{then} \\
& \quad PD^1(a, skP, B^a); bad \\
& \mathsf{else}\ out(A^b, g^b, A) \\
\| SF^0(b, skS); & \mathsf{if}\ A = g^a\ \mathsf{then} \\
& \quad out(k, g^b, A) \\
& \mathsf{else\ if}\ \neg T(B, \overline{s})\ \mathsf{then} \\
& \quad SF^1(b, skS, A^b); bad \\
& \mathsf{else}\ out(A^b, g^b, A)
\end{array}$$

Note that the k used here is a fresh name, which could be considered as a long term secret, i.e inside $\overline{p}$. We may prove this without considering P^1 and S^1, and replacing them by oracles which can simulate them. The proof of A-3 can once again be derived from the proof performed in Appendices D.3 and D.4. Note that here, the proof is greatly simplified because our modified

forwarding agent ensure that any signed session identifier is honest, and the secrecy of the name k is not even required to perform the proof. This proof could also performed without the additional check added to the forwarding agent, but it would then require a cryptographic assumption regarding the encryption.

And B-1:

$$PD^1(k'); Y(k') \| SF^1(k'); Z(k') \cong_{\mathcal{O}_{KE_1}, \mathcal{O}_{KE_2^k}} PD^1(k'); Y'(k') \| SF^1(k'); Z'(k')$$

With the oracles:

- $\mathcal{O}^k_{FPS}$ allows to simulate (A-1) the other honnest sessions of PD and SF, it corresponds to $\mathcal{O}^{\mathsf{sign}}_{T'_P, skS, \overline{s}}, \mathcal{O}^{\mathsf{sign}}_{T'_S, skP, \overline{s}}, \mathcal{O}_{a_i, b_i}$ of Appendix D.4.
- $\mathcal{O}_{YZ}$ allows to simulate (C-1) the continuation, i.e the protocols of the form $\mathtt{in}(k); PD^1(k); Y(k) \| \mathtt{in}(k); SF^1(k); Z(k)$
- $\mathcal{O}_{KE_2^k}$ allows to simulate (C-2) $!_{n^2} FA(k) \| PD(k) \| SF$ (it is similar to $\mathcal{O}_{FPS}$).

Here, we do not commit to any transport protocol used after the SSH key exchange. It would probably use some encryption using the fresh key. Then, if for instance IND-CCAis required to prove $Y \| Z \cong Y' \| Z'$, to prove $B-1$, we would need to assume that IND-CCAis still valid even when the attacker has access to the hash of a message containing the key used for encryption. This holds for instance in the random oracle model. A proof of $B-1$ could then be derived from the proof of $Y \| Z \cong Y' \| Z'$ which would still be valid under an oracle producing hashes of the key, i.e an oracle which could simulate PD^1 and SF^1.

F Proofs

F.1 Formal Corollary for Key Exchange

We denote $\overline{p} = \{id^I, id^R\}$ and $\overline{s} = \{lsid^I_i, lsid^R_i\}_{i \in \mathbb{N}}$ the set of all the copies of the local session identifiers.

Formalizing the previous Section, to prove the security of a key exchange, we can use the following Corollary of Theorem 5.

Corollary 1. *Let $\mathcal{O}_{ke}$, $\mathcal{O}$ be oracles and $KE_i[\__1, \__2] := I(lsid^I_i, id^I); \__1 \| R(lsid^R_i, id^R); \__2$ a key exchange protocol, such that I binds $x^I, x^I_{id}, x^I_{lsid}$, R binds $x^R, x^R_{id}, x^R_{lsid}$ and $\mathcal{N}_l(KE)$ is disjoint of the oracle support. Let id^I, id^R be names and $\overline{s}^I = \{lsid^I_i\}_{i \in \mathbb{N}}, \overline{s}^R = \{lsid^R_i\}_{i \in \mathbb{N}}$ sets of names :*

1. $\forall i \geq 1, (\nu lsid^I_i, id^I, lsid^R_i, id^R.$

   $$KE_i[\mathtt{out}(\langle x^I, lsid^I_i, x^I_{lsid}, x^I_{id} \rangle), \mathtt{out}(\langle x^R, lsid^R_i, x^R_{lsid}, x^R_{id} \rangle)] \| \mathtt{out}(\langle lsid^R_i, lsid^I_i \rangle)$$

   is $\mathcal{O}_{ke}$ simulatable)).

2. *$\overline{s}$ is disjoint of the support of $\mathcal{O}$.*

$$KE_0[\mathtt{out}(\langle x^I, lsid_0^I, x_{lsid}^I, x_{id}^I\rangle), \mathtt{out}(\langle x^R, lsid_0^R, x_{lsid}^R, x_{id}^R\rangle)] \cong_{\mathcal{O}_{ke},\mathcal{O}}$$
$$KE_0[\ \ \textit{if } x_{lsid}^I = lsid_0^R \wedge x_{id}^I = id^R \textit{ then}$$
$$\quad \mathtt{out}(\langle k, lsid_0^I, x_{lsid}, x_{id}\rangle)$$
$$\textit{else if } x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R \textit{ then}$$
$$\quad \perp$$
3. $$\textit{else } out(\langle x^I, lsid_0^I, x_{lsid}^I, x_{id}^I\rangle),$$
$$\textit{if } x_{lsid}^R = lsid_0^I \wedge x_{id}^R = id^I \textit{ then}$$
$$\quad \mathtt{out}(\langle k, lsid_0^R, x_{lsid}^R, x_{id}^R\rangle)$$
$$\textit{else if } x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I \textit{ then}$$
$$\quad \perp$$
$$\textit{else } \mathtt{out}(\langle x^R, lsid_0^R, x_{lsid}^R, x_{id}^R\rangle)]$$

Then, for any N which depends on the security parameter:

$$\|^{i\leq N} KE_i[\mathtt{out}(x^I), \mathtt{out}(x^R)] \cong_{\mathcal{O}}$$
$$\|^{i\leq N} \quad KE_i[\ \ \textit{if } (x_{id}^I = id^R) \textit{ then}$$
$$\textit{if}_{1\leq j\leq N} x_{lsid}^I = lsid_j^R \wedge x_{id}^I = id_R \textit{ then}$$
$$\mathtt{out}(k_{i,j})$$
$$\textit{else } \mathtt{out}(x^I),$$
$$\textit{if } (x_{id}^R = id_I) \textit{ then}$$
$$\textit{if}_{1\leq j\leq N} x_{lsid}^R = lsid_j^I \wedge x_{id}^R = id_I \textit{ then}$$
$$\mathtt{out}(k_{j,i})$$
$$\textit{else } \mathtt{out}(x^R)]$$

Then, building upon the previous Corollary and the sequential composition Theorems, the following Corollary shows the precise requirements to prove the security of a protocol which uses a key exchange, for an bounded number of session and with long term secrets shared between the key exchange and the protocol.

Corollary 2. *Let $\mathcal{O}_T$, $\mathcal{O}_{ke}$, $\mathcal{O}_r$,$\mathcal{O}_{P,Q}$ be oracles and*

$KE_i[\__1, \__2] := I(lsid_i^I, id^I);\__1 \| R(lsid_i^R, id^R);\__2$ a key exchange protocol, such that I binds $x^I, x_{id}^I, x_{lsid}^I$, R binds $x^R, x_{id}^R, x_{lsid}^R$ and $\mathcal{N}_l(KE)$ is disjoint of the oracle support. Let id^I, id^R be names, $\overline{s}^I = \{lsid_i^I\}_{i\in\mathbb{N}}$,$\overline{s}^R = \{lsid_i^R\}_{i\in\mathbb{N}}$ and $\overline{s} = \overline{s}^I \cap \overline{s}^R$ sets of names.

Let $\overline{p} = \{id^I, id^R\}$, $P(x, \overline{y}) = P_1(x, \overline{y}) \| P_2(x, \overline{y})$ and $Q(x, \overline{y}, \overline{z}) = Q_1(x, \overline{y}, \overline{z}) \| Q_1(x, \overline{y}, \overline{z})$ be parameterized protocols, such that $\mathcal{N}_l(P, Q)$ is disjoint of the oracle support.

I-1 $\forall i \geq 1, (\nu lsid_i^I, id^I, lsid_i^R, id^R.KE_i[\mathtt{out}(x^I), \mathtt{out}(x^R)] \| \mathtt{out}(\langle lsid_i^R, lsid_i^I\rangle)$ is $\mathcal{O}_T$-simulatable)).

I-2 $\overline{s}$ is disjoint of the support of $\mathcal{O}_{P,Q}$.

$$KE_0[\mathtt{out}(\langle x^I, lsid_0^I, x_{lsid}^I, x_{id}^I\rangle), \mathtt{out}(\langle x^R, lsid_0^R, x_{lsid}^R, x_{id}^R\rangle) \cong_{\mathcal{O}_T, \mathcal{O}_{P,Q}}$$

$$\begin{array}{ll} KE_0 & [\mathsf{if}\ x_{lsid}^I = lsid_0^R \wedge x_{id}^I = id^R\ \mathsf{then} \\ & \quad \mathtt{out}(\langle k, lsid_0^I, x_{lsid}^I, x_{id}^I\rangle) \\ & \mathsf{else\ if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathsf{then} \\ & \quad \bot \\ \text{I-3} & \mathsf{else}\ \mathtt{out}(\langle x^I, lsid_0^I, x_{lsid}^I, x_{id}^I\rangle), \\ & \mathsf{if}\ x_{lsid}^R = lsid^I \wedge x_{id}^R = id^I\ \mathsf{then} \\ & \quad \mathtt{out}(\langle k, lsid_0^R, x_{lsid}^R, x_{id}^R\rangle) \\ & \mathsf{else\ if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I\ \mathsf{then} \\ & \quad \bot \\ & \mathsf{else}\ \mathtt{out}(\langle x^R, lsid_0^R, x_{lsid}^R, x_{id}^R\rangle)] \end{array}$$

and

R-1 $\forall\ 1 \leq i, j \leq n, \nu\overline{p}, k_{i,j}.P_0(\overline{p}, k_{i,j})$ *is* $\mathcal{O}_r$*-simulatable.*

R-2 $\forall\ 1 \leq i \leq n, \nu\overline{p}, k_{i,j}.Q_0(\overline{p}, k_{i,j})$ *is* $\mathcal{O}_r$*-simulatable.*

R-3 $\overline{s}$ *is disjoint of the support of* $\mathcal{O}_k$*.*

R-4 $P_0(\overline{p}, k) \cong_{\mathcal{O}_r, \mathcal{O}_{ke}} Q_0(\overline{p}, k)$

and

C-1 $\nu\overline{p}.in(x_i^I).P_i^I(x_i^I) \| in(x_i^R).P_i^R(x_i^R)$ *is* $\mathcal{O}_{P,Q}$*-simulatable.*

$$\begin{array}{lll} \|^{i\leq n} & KE_i[& \mathsf{if}\ (x_{id}^I = id^R)\ \mathsf{then} \\ & & \underset{1\leq j\leq n}{\mathsf{if}}\ (x_{lsid}^I = lsid_j^R \wedge x_{id}^I = id_R)\ \mathsf{then} \\ & & \quad out(\langle i, j\rangle) \\ & & \mathsf{else}\ P_i^I(x_i^I), \\ \text{1. } \nu\overline{p}. & & \mathsf{if}\ (x_{id}^R = id_I)\ \mathsf{then} \qquad \text{is } \mathcal{O}_{ke}\text{-simulatable.} \\ & & \underset{1\leq j\leq n}{\mathsf{if}}\ (x_{lsid}^R = lsid_j^I \wedge x_{id}^R = id_I)\ \mathsf{then} \\ & & \quad out(\langle i, j\langle) \\ & & \mathsf{else}\ P_i^R(x_i^R)] \end{array}$$

Then, for any n which may depend on the security parameter:

$$\|^{i\leq n} KE_i[P_i^I(x_i^I), P_i^R(x_i^R)] \cong$$

$$\|^{i\leq n} KE_i[\mathsf{if}\ x_{id}^I = id^R\ \mathsf{then}\ Q_i^I(x_i^I)\ \mathsf{else}\ P_i^I(x_i^I), \mathsf{if}\ x_{id}^R = id^I\ \mathsf{then}\ Q_i^R(x_i^R)\ \mathsf{else}\ P_i^R(x_i^R)]$$

F.2 Formal Corollary for Key Confirmations

The Theorem for those key exchanges is very similar to Corollary 2. The main difference is that now, instead of working on a key exchange $KE := I(lsid^I, id^I)|R(lsid^R, id^R)[$, we further split I and R, in $I = I^0; I^1$ and $R := R^0; R^1$, where I^0 and R^0 will corresponds to the key exchange up to but not including the first use of the secret key, and I^1 and R^1 as the remainder of the protocol.

Corollary 3. *Let $\mathcal{O}_{KE}$, $\mathcal{O}_r$,$\mathcal{O}_{P,Q}$ be oracles and*

$$KE_i[\__1, \__2] := I_i(lsid_i^I, id^I); \__1 | R_i(lsid^R i, id^R); \__2$$

a key exchange protocol with $I_i(lsid_i^I, id^I) := I_i^0(lsid_i^I, id^I); I_i^1(x^I)$ and $R_i(lsid_i^R, id^R) := R_i^0(lsid_i^R, id^R); R_i^1(x^R)$ such that I^0 binds x^I, x_{id}, x_{lsid}, R^0 binds x^R, x_{id}, x_{lsid} and $\mathcal{N}_l(KE)$ is disjoint of the oracles support. Let $\overline{p} = \{id^I, id^R\}$, $P_i(x, \overline{y}) = P_i^I(x, \overline{y}) \| P_i^R(x, \overline{y})$,$Q(x, \overline{y}, \overline{z}) = Q_i^I(x, \overline{y}, \overline{z}) \| Q_i^R(x, \overline{y}, \overline{z})$, $C_i(\overline{z})$ and $D_i(\overline{z})$ be protocols, such that $\mathcal{N}_l(P, Q, C, D)$ is disjoint of the oracles support.

Let id^I, id^R be names, $\overline{s}^I = \{lsid_i^I\}_{i \in \mathbb{N}}$,$\overline{s}^R = \{lsid_i^R\}_{i \in \mathbb{N}}$ and $\overline{s} = \overline{s}^I \cap \overline{s}^R$ sets of names.

A-1 $\forall i \in \mathbb{N}, (\nu lsid_i^I, id^I, lsid_i^R, id^R.C_i(\overline{p}) \| I_i^0(lsid_i^I, id^I); \mathtt{out}(x^I) \| R_i^0(lsid_i^R, id^R); \mathtt{out}(x^R)$ *is $\mathcal{O}_{KE}$ simulatable)).*

A-2 $\overline{s}$ *is disjoint of the support of $\mathcal{O}_p$.*

A-3

$$\begin{array}{ll}
C_i(\overline{p}) \| I_0^0(lsid_0^I, id^I); & \textit{if } x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R \textit{ then} \\
& \quad I^1(x^I); \mathtt{out}(x^I) \\
& \textit{else } \mathtt{out}(\langle x^I, lsid_0^I, x_{lsid}^I, x_{id}^I \rangle) \\
\| R^0(lsid_0^R, id^R); & \textit{if } x_{lsid}^R \notin \overline{s}^I \wedge x_{id} = id^I \textit{ then} \\
& \quad R^1(x^R); \mathtt{out}(x^R) \\
& \textit{else } \mathtt{out}(\langle x^R, lsid^R, x_{lsid}^R, x_{id}^R \rangle) \\
\cong_{\mathcal{O}_{KE}, \mathcal{O}_p} & \\
C_i(\overline{p}) \| I^0(lsid_0^I, id^I); & \textit{if } x_{lsid}^I = lsid^R \wedge x_{id} = id^R \textit{ then} \\
& \quad \mathtt{out}(\langle k, lsid_0^I, x_{lsid}^I, x_{id}^I \rangle) \\
& \textit{else if } x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R \textit{ then} \\
& \quad I^1(x^R); \perp \\
& \textit{else } \mathtt{out}(\langle x^I, lsid^I, x_{lsid}^I, x_{id}^I \rangle) \\
\| R^0(lsid_0^R, id^R); & \textit{if } x_{lsid}^R = lsid^I \wedge x_{id}^R = id^I \textit{ then} \\
& \quad \mathtt{out}(\langle k, lsid_0^R, x_{lsid}^R, x_{id}^R \rangle) \\
& \textit{else if } x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I \textit{ then} \\
& \quad I^1(x^R); \perp \\
& \textit{else } \mathtt{out}(\langle x^R, lsid_0^R, x_{lsid}^R, x_{id}^R \rangle)
\end{array}$$

and for any N which may depend on the security parameter:

B-1 $\|^{i \leq N^2} D_i(\overline{p}) \| I_i^1(k_i); P_i^I(\overline{p}, k_i) \| B_i^1(k_i); P_i^R(\overline{p}, k_i) \cong_{\mathcal{O}_r, \mathcal{O}_k} \|^{i \leq n^2} D_i(\overline{p}) \| I_i^1(k_i); Q_i^I(\overline{p}, k_i) \| B_i^1(k_i); Q_i^R(\overline{p}, k_i)$

and

C-1 $\nu \overline{p}, lsid_i^I, lsid_i^R.D_i(\overline{p}) \| in(x).P_i(x) \ \| in(x).Q_i(x) \| in(x).I_i^1(x); P_i^I(x) \| in(x).R_i^1(x); P_i^R(x)$ $\| in(x).I_i^1(x); Q_i^I(x) \| in(x).R_i^1(x); Q_i^R(x)$ *is $\mathcal{O}_p$ simulatable.*

$$\|^{i\leq N} \quad C_i(\overline{p})\|I_i^0(lsid_i^I, id^I); \quad \begin{array}{l} \mathsf{if} \underset{1\leq j\leq N}{} x_{lsid}^I = lsid_j^R \wedge x_{id}^I = id^R \ \mathsf{then} \\ \quad \mathtt{out}(\langle i,j\rangle) \\ \mathsf{else\ if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R \ \mathsf{then} \\ \quad I_i^1(x^I); \bot \\ \mathsf{else}\ I_i^1(x^I); P_i^I(x^I) \end{array}$$

C-2 $\nu\overline{p}$.

$$\|R_i^0(lsid_i^R, id^R)[\quad \begin{array}{l} \mathsf{if} \underset{1\leq j\leq N}{} x_{lsid}^R = lsid_j^I \wedge x_{id}^R = id^I \ \mathsf{then} \\ \quad \mathtt{out}(\langle i,j\rangle) \\ \mathsf{else\ if}\ (x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I \ \mathsf{then} \\ \quad R_i^1(x^R); \bot \\ \mathsf{else}\ R_i^1(x^R); P_i^R(x^R) \end{array}$$

is $\mathcal{O}_k$ simulatable.

Then, for any n:

$$\|^{i\leq N} C_i(\overline{p})\|D_i(\overline{p})\|KE_i[P_i^I(x^I), P_i^R(x^R)] \cong$$
$$\|^{i\leq N} C_i(\overline{p})\|D_i(\overline{p})\|KE_i[\mathsf{if}\ x_{id}^I = id^R\ \mathsf{then}\ Q_i^I(x^I)\ \mathsf{else}\ P_i^I(x^I), \mathsf{if}\ x_{id}^R = id^I\ \mathsf{then}\ Q_i^R(x^R)\ \mathsf{else}\ P_i^R(x^R)]$$

F.3 Oracle Simulation

We first show that $\mathcal{O}$-simulation, whose definition implies the identical distributions of two messages produced either by the simulator of by the oracle, implies the equality of distributions of message sequences produced by either the oracle or the simulator.

Lemma 9. *Given a cryptographic library $\mathcal{M}^f$, a sequence of names $\overline{n}$, an oracle $\mathcal{O}$ with support $\overline{n}$ and a protocol P, that is* $\mathcal{O}$-simulatable *with $\mathcal{A}^{\mathcal{O}}$, we have, for every $\overline{x}, \overline{y}, c, r_2, r_{\mathcal{B}} \in \{0,1\}^\star$, every $\overline{v} \in D_{\overline{n}}^\eta$, for every $m \geq 1$, for every PTOM $\mathcal{B}^{\mathcal{O}}$ (using tags prefixed by 1):*

$$\begin{array}{l} \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta_m^1 = \overline{x}, \phi_m^1 = \overline{y} |\ [\![\overline{n}]\!]_{\rho_s}^\eta = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\ \quad = \quad \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta_m^2 = \overline{x}, \phi_m^2 = \overline{y} |\ [\![\overline{n}]\!]_{\rho_s}^\eta = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \end{array}$$

where we split $\rho_{\mathcal{O}}$ into $\rho_{\mathcal{O}}^{\mathcal{A}} \uplus \rho_{\mathcal{O}}^{\mathcal{B}}$ such that $\mathcal{O}$ called by $\mathcal{B}$ only accesses $\rho_{\mathcal{O}}^{\mathcal{B}}$ and $\mathcal{O}$ called by $\mathcal{A}$ only accesses $\rho_{\mathcal{O}}^{\mathcal{A}}$ (which is possible thanks to the distinct prefixes).

Proof. We proceed by induction on m. Let us fix $\overline{x}, \overline{y}, c, r_2, r_{\mathcal{B}} \in \{0,1\}^\star$ and $\overline{v} \in D_{\overline{n}}^\eta$. We assume that:

$$\begin{array}{l} \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta_m^1 = \overline{x}, \phi_m^1 = \overline{y} |\ [\![\overline{n}]\!]_{\rho_s}^\eta = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\ \quad = \quad \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta_m^2 = \overline{x}, \phi_m^2 = \overline{y} |\ [\![\overline{n}]\!]_{\rho_s}^\eta = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \end{array}$$

We define $v_{m+1}^i = \mathcal{B}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\mathcal{M}_f, \rho_{r_2}, \eta, \phi_m^i)$.
As the support of $\mathcal{O}$ is $\overline{n}$, we have that $\mathcal{O}(\rho_s, \rho_{\mathcal{O}}) = \mathcal{O}(\pi_{\overline{k}}(\rho_s, \eta), \rho_{\mathcal{O}})$.

Using conditional probabilities, we have that:

$$\begin{array}{l} \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta_{m+1}^1 = \overline{x}, \phi_m^1 = \overline{y} |\ [\![\overline{n}]\!]_{\rho_s}^\eta = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\ \quad = \quad \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{v_{m+1}^1 = x_{m+1} |\ \theta_m^1 = \overline{x}, \phi_m^2 = \overline{y}, [\![\overline{n}]\!]_{\rho_s}^\eta = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\ \qquad \times \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta_m^1 = \overline{x}, \phi_m^1 = \overline{y} |\ [\![\overline{n}]\!]_{\rho_s}^\eta = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \end{array}$$

Now, if we define $\mathcal{O}_{\overline{v},r_{\mathcal{B}}}$ such that $\mathcal{O}_{\overline{v},r_{\mathcal{B}}} = \mathcal{O}(\pi_{\overline{n}}(\rho_s,\eta),\rho^{\mathcal{B}}_{\mathcal{O}})$ when $[\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}$ and $\rho^{\mathcal{B}}_{\mathcal{O}} = r_{\mathcal{B}}$, we have that

$$\begin{aligned}
&\mathbb{P}_{\rho_s,,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{v^1_{m+1} = x_{m+1} |\, \theta^1_m = \overline{x}, \phi^1_m = \overline{y}, [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}, \rho^{\mathcal{B}}_{\mathcal{O}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\}\\
&\quad =^1 \quad \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{B}^{\mathcal{O}(\pi_{\overline{n}}(\rho_s,\eta),\rho^{\mathcal{B}}_{\mathcal{O}})}(\mathcal{M}_f,\rho_{r_2},\eta,\phi^1_m) = x_{m+1}\\
&\qquad\qquad | \,\theta^1_m = \overline{x}, \phi^1_m = \overline{y}, [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}, \rho^{\mathcal{B}}_{\mathcal{O}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\}\\
&\quad =^2 \quad \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{B}^{\mathcal{O}_{\overline{v},r_{\mathcal{B}}}}(\mathcal{M}_f,r_2,\eta,\overline{y}) = x_{m+1}\\
&\qquad\qquad | \,\theta^1_m = \overline{x}, \phi^1_m = \overline{y}, [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}, \rho^{\mathcal{B}}_{\mathcal{O}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\}\\
&\quad =^3 \quad \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{B}^{\mathcal{O}_{\overline{v},r_{\mathcal{B}}}}(\mathcal{M}_f,r_2,\eta,\overline{y}) = x_{m+1}\}\\
&\quad =^4 \quad \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{B}^{\mathcal{O}_{\overline{v},r_{\mathcal{B}}}}(\mathcal{M}_f,r_2,\eta,\overline{y}) = x_{m+1}\\
&\qquad\qquad | \,\theta^2_m = \overline{x}, \phi^2_m = \overline{y}, [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}, \rho^{\mathcal{B}}_{\mathcal{O}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\}\\
&\quad =^5 \quad \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{B}^{\mathcal{O}(\pi_{\overline{n}}(\rho_s,\eta),\rho^{\mathcal{B}}_{\mathcal{O}})}(\mathcal{M}_f,\rho_{r_2},\eta,\phi^2_m) = x_{m+1}\\
&\qquad\qquad | \,\theta^2_m = \overline{x}, \phi^2_m = \overline{y}, [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}, \rho^{\mathcal{B}}_{\mathcal{O}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\}
\end{aligned}$$

Justified with:

1. because $\mathcal{O}(\rho_s,\rho_{\mathcal{O}}) = \mathcal{O}(\pi_{\overline{n}}(\rho_s,\eta),\rho^{B}_{\mathcal{O}})$;
2. $\mathcal{O}(\pi_{\overline{n}}(\rho_s,\eta),\rho^{B}_{\mathcal{O}}) = \mathcal{O}_{\overline{v},r_{\mathcal{B}}}$, and $\phi^1_m = \overline{y}$;
3. the considered event does not depends on any of the conditional events removed;
4. the considered event does not depends on any of the conditional events added;
5. reversing the previous steps.

So we conclude that, as we also have the induction hypothesis:

$$\begin{aligned}
&\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta^1_{m+1} = \overline{x}, \phi^1_m = \overline{y} |\, [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}, \rho^{\mathcal{B}}_{\mathcal{O}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\}\\
&\quad = \quad \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta^2_{m+1} = \overline{x}, \phi^2_m = \overline{y} |\, [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}, \rho^{\mathcal{B}}_{\mathcal{O}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \quad (i)
\end{aligned}$$

We now define:

$$u^1_{m+1} = \mathcal{A}^{\mathcal{O}(\pi_{\overline{n}}(\rho_s,\eta),\rho^{\mathcal{A}}_{\mathcal{O}})}(\mathcal{M}_f,\rho_{r_1},\theta^1_m \uplus v^1_{m+1},\eta)$$

$$u^2_{m+1} = \mathcal{O}_P(\rho_s,\theta^2_m \uplus v^2_{m+1})$$

We define the Turing machine $\underline{\mathcal{B}}$, such that:

$$\begin{aligned}
&\underline{\mathcal{B}}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\mathcal{M}_f,\rho_{r_2},\eta,\phi^i_m) :=\\
&\quad \mathsf{if}\ \forall j \leq m+1, \mathcal{B}^{\mathcal{O}(\overline{v},r_{\mathcal{B}})}(\mathcal{M}_f,r_2,\eta,\phi^i_j) = x_j\\
&\qquad \wedge \phi^i_m = \overline{y}\\
&\quad \mathsf{then}\ \mathcal{B}^{\mathcal{O}(\overline{v},r_{\mathcal{B}})}(\mathcal{M}_f,r_2,\eta,\phi^i_m)\\
&\quad \mathsf{else}\ \bot
\end{aligned}$$

We then define v'_m and θ'_m for $\underline{\mathcal{B}}$ similarly as v_m for $\mathcal{B}$.

We define $\mathcal{O}_{\overline{v},\rho_{\mathcal{O}}^{\mathcal{A}}}$ such that $\mathcal{O}_{\overline{v},\rho_{\mathcal{O}}^{\mathcal{A}}} = \mathcal{O}(\pi_{\overline{n}}(\rho_s,\eta),\rho_{\mathcal{O}}^{\mathcal{A}})$ when $[\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}$. We then have:

$$
\begin{aligned}
&\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{u_{m+1}^1 = y_{m+1} |\, \theta_{m+1}^1 = \overline{x}, \phi_m^1 = \overline{y}, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\
&\quad =^1 \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}_{\overline{v},\rho_{\mathcal{O}}^{\mathcal{A}}}}(\mathcal{M}_f,\rho_{r_1},\overline{x},\eta) = y_{m+1} |\, \theta_{m+1}^1 = \overline{x}, \phi_m^1 = \overline{y}, \\
&\qquad\qquad [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\
&\quad =^2 \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}_{\overline{v},\rho_{\mathcal{O}}^{\mathcal{A}}}}(\mathcal{M}_f,\rho_{r_1},\theta_{m+1}^{1'},\eta) = y_{m+1} |\, \theta_{m+1}^1 = \overline{x}, \phi_m^1 = \overline{y}, \\
&\qquad\qquad [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\
&\quad =^3 \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}_{\overline{v},\rho_{\mathcal{O}}^{\mathcal{A}}}}(\mathcal{M}_f,\rho_{r_1},\theta_{m+1}^{1'},\eta) = y_{m+1} |\, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\} \\
&\qquad \times (\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta_{m+1}^1 = \overline{x}, \phi_m^1 = \overline{y} |\, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\
&\qquad\quad \times \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2 |\, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\})^{-1} \\
&\quad =^4 \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{O}_P(\rho_s,\theta_{m+1}^{2'}) = y_{m+1} |\, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\} \\
&\qquad \times (\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta_{m+1}^1 = \overline{x}, \phi_m^1 = \overline{y}, |\, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\
&\qquad\quad \times \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2 |\, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\})^{-1} \\
&\quad =^5 \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{O}_P(\rho_s,\theta_{m+1}^{2'}) = y_{m+1} |\, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\} \\
&\qquad \times (\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta_{m+1}^2 = \overline{x}, \phi_m^2 = \overline{y}, |\, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\
&\qquad\quad \times \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2 |\, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}\})^{-1} \\
&\quad =^6 \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{O}_P(\rho_s,\theta_{m+1}^{2'}) = y_{m+1} |\, \theta_{m+1}^2 = \overline{x}, \phi_m^2 = \overline{y}, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}, \\
&\qquad\qquad \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\
&\quad =^7 \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{O}_P(\rho_s,\overline{x}) = y_{m+1} |\, \theta_{m+1}^2 = \overline{x}, \phi_m^2 = \overline{y}, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}, \\
&\qquad\qquad \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\
&\quad =^8 \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{u_{m+1}^2 = y_{m+1} |\, \theta_{m+1}^2 = \overline{x}, \phi_m^2 = \overline{y}, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}, \\
&\qquad\qquad \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\}
\end{aligned}
$$

Justified with:

1. using the conditional probabilities;
2. by definition of $\underline{\mathcal{B}}$ which produces $\overline{x}$ under the conditional events;
3. using conditional probabilities, as $\theta_m \neq \overline{x} \vee \phi_m \neq \overline{y} \Rightarrow \underline{\mathcal{B}} = \bot$;
4. by $\mathcal{O}$ simulatability on $\underline{\mathcal{B}}$;
5. using (i);
6. using conditional probabilities, as $\theta_m \neq \overline{x} \vee \phi_m \neq \overline{y} \Rightarrow \underline{\mathcal{B}} = \bot$;
7. by definition of $\underline{\mathcal{B}}$ which produces $\overline{x}$ under the conditional events;
8. using the conditional probabilities.

Combining the previous equality with equation (i) finally yields through conditional probabilities:

$$
\begin{aligned}
&\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta_{m+1}^1 = \overline{x}, \phi_{m+1}^1 = \overline{y} |\, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \\
&\quad = \quad \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\theta_{m+1}^2 = \overline{x}, \phi_{m+1}^2 = \overline{y} |\, [\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}, \rho_{\mathcal{O}}^{\mathcal{B}} = r_{\mathcal{B}}, \rho_{r_2} = r_2\}
\end{aligned}
$$

□

F.4 Autocomposition Results

Proposition 18. *Let $\mathcal{O}$ be an oracle, two parameterized processes $P(x), Q(x)$, a set of names $\overline{n} = \mathcal{N}_g(P, Q)$ and fresh names k_0, l. We assume that $\mathcal{N}_l(P, Q)$ is disjoint of the support of $\mathcal{O}$. If:*

- *$\nu\overline{n}.\mathtt{in}(c_P, x); P(x) \| \mathtt{in}(c_Q, x); Q(x)$ is $\mathcal{O}$-simulatable, and*
- *$P(k_0); \mathtt{out}(c_P, x) \| Q(k_0); \mathtt{out}(c_Q, x) \cong_{\mathcal{O}} P(k_0); \mathtt{out}(c_P, l) \| Q(k_0); \mathtt{out}(c_Q, l)$*

then, for any N,

$$\begin{aligned} &P(k_0); P(x)^{;N}; \mathtt{out}(c_P, x) \| Q(k_0); Q(x)^{;N}; \mathtt{out}(c_Q, x) \\ &\quad \cong_{\mathcal{O}} P(k_0); P(x)^{;N}; \mathtt{out}(c_P, l) \| Q(k_0); Q(x)^{;N}; \mathtt{out}(c_Q, l) \end{aligned}$$

Proof. We proceed by induction on N. The result is exactly the first hypothesis for $N = 0$.

Given some $N > 1$, we assume that

$$P(k_i)^{;N-1}; out(k) \| Q(k_i)^{;N-1}; out(k) \cong_{\mathcal{O}} P(k_i)^{;N-1}; out(l) | Q(k_i)^{;N-1}; out(l) \quad (i)$$

In the following, we will write $P(k_i)^{;N-1}$ for $P(k_i); P(k)^{;N-2}$ and we will omit to mention the α-renaming made over the local names in $\mathcal{N}_l(P, Q)$ between the different copies of P and Q. The renaming is however essential so that we may for instance have $\mathcal{N}_l(P^{N-1}(k)) \cap \mathcal{N}_l(P) = \emptyset$ when we wish to apply Theorem 4. This silent renaming is possible because $\mathcal{N}_l(P, Q)$ is not contained in the support of $\mathcal{O}$.

We obtain by application of Theorem 4 with $A = P(k_i)^{;N-1}$, $B = Q(k_i)^{;N-1}$, $P_1(x) := P(x)^{;0}; out(k)$ and $P_2(x) := Q(x)^{;0}; out(k)$:

$$P^{;N}(k_i); out(k) \| Q^{;N}(k_i); out(k) \cong_{\mathcal{O}} P^{;N-1}(k_i); P(l)^{;0}; out(k) \| Q^{;N-1}(k_i); Q(l)^{;0}; out(k) \quad (I)$$

Now, with Theorem 2 applied on $P(l)^{;0}; out(k) \| Q(l)^{;0}; out(k) \cong_{\mathcal{O}} P(l)^{;0}; out(l') \| Q(l)^{;0}; out(l')$ with l' a fresh name, with $P := P(k_i)^{;N-1}$ and $Q := Q(k_i)^{;N-1}$, we obtain:

$$P(k_i)^{;N-1}; P(l)^{;0}; out(k) \| Q^{;N-1}(k_i); Q(l); out(k) \cong_{\mathcal{O}} P(k_i)^{;N-1}; P(l)^{;0}; out(l') \| Q(k_i)^{;N-1}; Q(l)^{;0}; out(l') \quad (II)$$

We also perform an application of Theorem 4 on (i) with $A = P(k_i)^{;N-1}$, $B = Q(k_i)^{;N-1}$, $P_1(k) := P(k_i)^{;0}; out(l)$ and $P_2(k) := Q(k_i)^{;0}; out(l)$:

$$P(k_i)^{;N-1}; P(l)^{;0}; out(l') | Q(k_i)^{;N-1}; P(l)^{;0}; out(l') \cong_{\mathcal{O}} P(k_i)^{;N}; out(l) \| Q(k_i)^{;N}; out(l) \quad (III)$$

We conclude by transitivity with (I),(II) and (III). □

Simulatability is stable by binding names that do not appear in the protocol, which means that we will be able simulate at the same times two simulatable protocol who do not share long term secret.

Lemma 37. *Given a cryptographic library $\mathcal{M}_f$, a sequence of names $\overline{n}$, an oracle $\mathcal{O}$ with support $\overline{n}$ and a sequence of terms $\overline{t}$, if $\nu\overline{n}.\overline{t}$ is* $\mathcal{O}$-simulatable *, then for any sequence of names $\overline{m}$ such that $\overline{m} \cap \mathcal{N}(t_1, \ldots, t_n) = \emptyset$, $\nu\overline{n} \cup \overline{m}.\overline{t}$ is* $\mathcal{O}$-simulatable.

Proof. Let there be a cryptographic library $\mathcal{M}_f$, a sequence of names $\overline{n}$, an oracle $\mathcal{O}$ with support $\overline{n}$ and a sequence of terms $\bar{t}$ $\mathcal{O}$*-simulatable.* As the names of $\overline{m}$ do not appear in $\bar{t}$, the probability of any event regarding $\bar{t}$ is independent from an event regarding $\overline{m}$ so we have for any PTOM $\mathcal{A}^{\mathcal{O}}$, η, sequences $\overline{c}, \overline{v}, \overline{w} \in \{0,1\}^*$,

$$\begin{aligned}
&\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\mathcal{M}_f, m_1, \ldots, m_k, \rho_{r_2}, \eta) = \overline{c} \mid [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}, [\![\overline{m}]\!]^{\eta}_{\rho_s} = \overline{w}\}\\
&\mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}(\mathcal{M}_f, m_1, \ldots, m_k, \rho_{r_2}, \eta) = \overline{c} \mid [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}\}\\
&\quad = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{[\![t_1, \ldots, t_n]\!]^{\eta}_{\rho_s,\rho_r,\rho_{\mathcal{O}}} = \overline{c} | [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}\}\\
&\quad = \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{[\![t_1, \ldots, t_n]\!]^{\eta}_{\rho_s,\rho_r,\rho_{\mathcal{O}}} = \overline{c} | [\![\overline{n}]\!]^{\eta}_{\rho_s} = \overline{v}, [\![\overline{m}]\!]^{\eta}_{\rho_s} = \overline{w}\}
\end{aligned}$$

Thus $\nu\overline{n} \cup \overline{m}.\bar{t}$ is $\mathcal{O}$*-simulatable.* □

Proposition 17. *Let $\mathcal{O}_r$ be an oracle parameterized by a sequence of names $\overline{s}$, and $\mathcal{O}$ an oracle. Let $\overline{p}$ be a sequence of names, $P(\overline{x})$, $R_i^1(\overline{x},\overline{y}), \ldots, R_i^k(\overline{x},\overline{y})$ and $Q(\overline{x})$ be protocols, such that $\mathcal{N}_l(R_i^1, \ldots, R_i^k)$ is disjoint of the oracle support. If we have, for sequences of names $\overline{lsid}^1, \ldots, \overline{lsid}^k$, with $\overline{s} = \{\overline{lsid}_i^j\}_{1 \leq j \leq k, i \in \mathbb{N}}$:*

1. *$\forall i, j \in \mathbb{N}, \nu\overline{p}, \overline{lsid}_i^j.R_i^j(\overline{p}, \overline{lsid}_i^j)$ is $\mathcal{O}_r$-simulatable.*
2. *$P(\overline{p}) \cong_{\mathcal{O}_r} Q(\overline{p})$*
3. *$\overline{s}$ is disjoint of the support of $\mathcal{O}$.*

Then, for any integers $N_1, \ldots, N_k$:

$$\begin{aligned}
&P(\overline{p}) \|^{i \leq N_1} (R_i^1(\overline{p}, \overline{lsid}_i^1)\| \ldots \|^{i \leq N_k} R_i^k(\overline{p}, \overline{lsid}_i^k)\\
&\quad \cong_{\mathcal{O},\mathcal{O}_r} Q(\overline{p}) \|^{i \leq N_1} R_i^1(\overline{p}, \overline{lsid}_i^1)\| \ldots \|^{i \leq N_k} R_i^k(\overline{p}, \overline{lsid}_i^k)
\end{aligned}$$

Specifically, there exists a polynomial p_S (independent of all R^j) such that if p_{R^j} is the polynomial bound on the runtime of the simulator for R^j, we have,

$$\begin{aligned}
&\mathrm{Adv}^{P(\overline{p}) \|^{i \leq N_1} (R_i^1(\overline{p}, \overline{lsid}_i^1)\| \ldots \|^{i \leq N_k} R_i^k(\overline{p}, \overline{lsid}_i^k) \cong_{\mathcal{O}} Q(\overline{p}) \|^{i \leq N_1} R_i^1(\overline{p}, \overline{lsid}_i^1)\| \ldots \|^{i \leq N_k} R_i^k(\overline{p}, \overline{lsid}_i^k)}(t)\\
&\quad \leq \mathrm{Adv}^{P(\overline{p}) \cong_{\mathcal{O},\mathcal{O}_r} Q(\overline{p})}\Big(p_S\big(t, N_1, |R^1|, \ldots, N_k, |R^k|, p_{R^1}(t), \ldots, p_{R^k}(t)\big)\Big)
\end{aligned}$$

Rather than proving the previous Theorem, where we recall that the protocols may depend on a predicate $T(x)$ whose interpretation depends on $\overline{s}$, we prove the version where P directly depends on $\overline{s}$.

Proposition 38. *Let $\mathcal{O}_r$ be an oracle parameterized by a sequence of names $\overline{s}$. Let $\overline{p}$ be a sequence of names, $P(\overline{x})$, $R_i^1(\overline{x},\overline{y},z), \ldots, R_i^k(\overline{x},\overline{y},z)$ and $Q(\overline{x},y)$ be protocols, such that $\mathcal{N}_l(R_i^1, \ldots, R_i^k)$ is disjoint of the oracle support. If we have, for sequences of names $\overline{lsid}^1, \ldots, \overline{lsid}^k$, with $\overline{s} = \{\overline{lsid}_i^j\}_{i,j \in \mathbb{N}}$:*

1. *$\forall i, j \in \mathbb{N}, \nu\overline{p}, \overline{lsid}_i^j.R_i^j(\overline{p}, \overline{lsid}_i^j, \overline{s})$ is $\mathcal{O}_r$-simulatable.*
2. *$P(\overline{p}) \cong_{\mathcal{O}} Q(\overline{p}, \overline{s})$*

Then, for any integers $N_1, \ldots, N_k$:

$$P(\overline{p})\|^{i\leq N_1}(R_i^1(\overline{p},\overline{lsid}_i^1,\overline{s})\| \ldots \|^{i\leq N_k} R_i^k(\overline{p},\overline{lsid}_i^k,\overline{s}) \\ \cong_{\mathcal{O}_r} Q(\overline{p},\overline{s})\|^{i\leq N_1} R_i^1(\overline{p},\overline{lsid}_i^1,\overline{s})\| \ldots \|^{i\leq N_k} R_i^k(\overline{p},\overline{lsid}_i^k,\overline{s})$$

Specifically, there exists polynomial p_S (independent of all R^j)such that if p_{R^j} is the polynomial bound on the runtime of the simulator for R^j, we have,

$$\mathrm{Adv}^{P(\overline{p})\|^{i\leq N_1}(R_i^1(\overline{p},\overline{lsid}_i^1,\overline{s})\|\ldots\|^{i\leq N_k}R_i^k(\overline{p},\overline{lsid}_i^k,\overline{s})\cong_{\mathcal{O}_r}Q(\overline{p},\overline{s})\|^{i\leq N_1}R_i^1(\overline{p},\overline{lsid}_i^1,\overline{s})\|\ldots\|^{i\leq N_k}R_i^k(\overline{p},\overline{lsid}_i^k,\overline{s})}(t) \\ \leq \mathrm{Adv}^{P(\overline{p})\cong_{\mathcal{O}}Q(\overline{p},\overline{s})}\Big(p_S\big(t,N_1,|R^1|,\ldots,N_k,|R^k|,p_{R^1}(t),\ldots,p_{R^k}(t)\big)\Big)$$

Proof. We prove the result for $k=1$, denoting R^1 as R, as the generalization is immediate. Let there be an integer n.

Hypothesis 1 with Lemma 37 gives us that for $1 \leq i \leq N$, $\nu lsid_i, \overline{p}.R_i(\overline{p}, lsid_i, \overline{s})$ is $\mathcal{O}_R$-simulatable.

Moreover, with $\delta = \{\overline{p}, \overline{s}\}$, $\mathcal{N}(R_i(\overline{p}, lsid_i, \overline{s})) \cap \delta = \{\overline{p}, lsid_i\}$, so thanks to Theorem 1, for $1 \leq i \leq N$, $\nu\delta.R(\overline{p}, lsid_i, \overline{s})$ is $\mathcal{O}_R$-simulatable.

Now, up to renaming of the local names of R (which is possible as they do not appear in the oracle support), we have that $\forall 1 \leq i < j \leq N. \mathcal{N}(R_i(\overline{p}, lsid_i, \overline{s})) \cap \mathcal{N}(R_j(\overline{p}, lsid_j, \overline{s})) \subset \delta$, so with Theorem 1 we have that $\|^{i\leq N} R_i(\overline{p}, lsid_i, \overline{s})$ is $\mathcal{O}_R$-simulatable.

Note that if R is simulatable by a simulator bounded by a polynomial $p_R(t)$ on an input of size t, then $\|i \leq N R(\overline{p}, lsid_i, \overline{s})$ is simulatable by a simulator bounded by a polynomial $q(n, p_R(t))$, where q is uniform in n and R.

Finally, we have that $\|^{i\leq N} R_i(\overline{p}, \overline{lsid_i}, \overline{s})$ is $\mathcal{O}_R$-simulatable and $P(\overline{p}, lsid_n) \cong_{\mathcal{O}} Q(\overline{p}, \overline{s})$, so we conclude with Theorem 2.

Instantiating the bound on the advantage from Theorem 2 with $|C| = n|R|$ and $p_C(t) = q(n, p_R(t))$ yields the desired result. □

Theorem 5. *Let $\mathcal{O}_r$, $\mathcal{O}$ be oracles both parameterized by a sequence of names $\overline{s}$. Let $\overline{p}$ be a sequence of names, $P_i(\overline{x}, \overline{y})$ and $Q_i(\overline{x}, \overline{y})$ be parameterized protocols, such that $\mathcal{N}_l(P, Q)$ is disjoint of the oracles support. If we have, for sequences of names $\overline{lsid}^P, \overline{lsid}^Q$, with $\overline{s} = \{\overline{lsid}_i^P, \overline{lsid}_i^Q\}_{i\in\mathbb{N}}$:*

1. *$\forall\ i \geq 1, \nu\overline{p}, \overline{lsid}_i^P.P_i(\overline{p}, \overline{lsid}_i^P)$ is $\mathcal{O}_r$-simulatable.*
2. *$\forall\ i \geq 1, \nu\overline{p}, \overline{lsid}_i^Q.Q_i(\overline{p}, \overline{lsid}_i^Q)$ is $\mathcal{O}_r$-simulatable.*
3. *$\overline{s}$ is disjoint of the support of $\mathcal{O}$.*
4. *$P_0(\overline{p}, \overline{lsid}_0^P) \cong_{\mathcal{O}_r,\mathcal{O}} Q_0(\overline{p}, \overline{lsid}_0^Q)$*

then,

$$\|^i P_i(\overline{p}, \overline{lsid}_i^P) \cong_{\mathcal{O}} \|^i Q_i(\overline{p}, \overline{lsid}_i^Q)$$

We once again generalize with the explicit dependence in $\overline{s}$.

Theorem 7. *Let $\mathcal{O}_r$, $\mathcal{O}$ be oracles both parameterized by a sequence of names $\overline{s}$. Let $\overline{p}$ be a sequence of names, $P_i(\overline{x}, \overline{y})$ and $Q_i(\overline{x}, \overline{y}, z)$ be parameterized protocols, such that $\mathcal{N}_l(P, Q)$ is disjoint of the oracles support. If we have, for sequences of names $\overline{lsid}^P, \overline{lsid}^Q$, with $\overline{s} = \{\overline{lsid}_i^P, \overline{lsid}_i^Q\}_{i\in\mathbb{N}}$:*

1. $\forall\ i \geq 1, \nu\overline{p}, \overline{lsid}_i^P.P_i(\overline{p}, \overline{lsid}_i^P)$ *is* $\mathcal{O}_r$*-simulatable.*

2. $\forall\ i \geq 1, \nu\overline{p}, \overline{lsid}_i^Q.Q_i(\overline{p}, \overline{lsid}_i^Q, \overline{s})$ *is* $\mathcal{O}_r$*-simulatable.*

3. $\overline{s}$ *is disjoint of the support of* $\mathcal{O}$*.*

4. $P_0(\overline{p}, \overline{lsid}_0^P) \cong_{\mathcal{O}_r,\mathcal{O}} Q_0(\overline{p}, \overline{lsid}_0^Q, \overline{s})$

then, $||^i P_i(\overline{p}, \overline{lsid}_i^P) \cong_{\mathcal{O}} ||^i Q_i(\overline{p}, \overline{lsid}_i^Q, \overline{s})$

Proof. By application of Theorem 5, we get that for all n_1, n_2,

$$\begin{gathered} P_0(\overline{p}, \overline{lsid}_0^P) \| ^{1<i\leq N_1} P_i(\overline{p}, \overline{lsid}_i^P) \| ^{1<i\leq N_2} Q_i(\overline{p}, \overline{s}, \overline{lsid}_i^Q) \\ \cong_{\mathcal{O}_r,\mathcal{O}} \\ Q_0(\overline{p}, \overline{s}, \overline{lsid}_0^Q) \| ^{1<i\leq N_1} P_i(\overline{p}, \overline{lsid}_i^P) \| ^{1<i\leq N_2} Q_i(\overline{p}, \overline{s}, \overline{lsid}_i^Q) \end{gathered}$$

By weakening of the attacker, we get:

$$\begin{gathered} P_0(\overline{p}, \overline{lsid}_0^P) \| ^{1<i\leq N_1} P_i(\overline{p}, \overline{lsid}_i^P) \| ^{1<i\leq N_2} Q_i(\overline{p}, \overline{s}, \overline{lsid}_i^Q) \\ \cong_{\mathcal{O}} \\ Q_0(\overline{p}, \overline{s}, \overline{lsid}_0^Q) \| ^{1<i\leq N_1} P_i(\overline{p}, \overline{lsid}_i^P) \| ^{1<i\leq N_2} Q_i(\overline{p}, \overline{s}, \overline{lsid}_i^Q) \end{gathered}$$

Then, for a polynomial p (assumed without loss of generality increasing), any $n = p(\eta)$, and all $j < n$:

$$\begin{gathered} P_0(\overline{p}, \overline{lsid}_0^P) \| ^{1<i\leq j-1} P_i(\overline{p}, \overline{lsid}_i^P) \| ^{1<i\leq N-j-1} Q_i(\overline{p}, \overline{s}, \overline{lsid}_i^Q) \\ \cong_{\mathcal{O}} \\ Q_0(\overline{p}, \overline{s}, \overline{lsid}_0^Q) \| ^{1<i\leq j-1} P_i(\overline{p}, \overline{lsid}_i^P) \| ^{1<i\leq N-j-1} Q_i(\overline{p}, \overline{s}, \overline{lsid}_i^Q) \end{gathered}$$

Through the renaming of the $\overline{lsid}$, which is possible as $\overline{s}$ is disjoint from the oracle support, we get that:

$$\begin{gathered} P_j(\overline{p}, \overline{lsid}_j^P) \| P_0(\overline{p}, \overline{lsid}_0^P) \ldots \| P_{j-1}(\overline{p}, \overline{lsid}_{j-1}^P) \| Q_{j+1}(\overline{p}, \overline{s}, \overline{lsid}_{j+1}^Q) \| \ldots Q_n(\overline{p}, \overline{s}, \overline{lsid}_n^Q) \\ \cong_{\mathcal{O}} \\ Q_j(\overline{p}, \overline{lsid}_j^Q, \overline{s}) \| P_0(\overline{p}, \overline{lsid}_0^P) \ldots \| P_{j-1}(\overline{p}, \overline{lsid}_{j-1}^P) \| Q_{j+1}(\overline{p}, \overline{s}, \overline{lsid}_{j+1}^Q) \| \ldots Q_n(\overline{p}, \overline{s}, \overline{lsid}_n^Q) \end{gathered}$$

Thanks to Theorem 5, there exist polynomial p_S such that, if p_P and p_Q are the polynomial bound on the runtime of the simulators for P or Q, for all j, we have that the advantage of any attacker running in time t against the previous indistinguishability, denoted D, is bounded by:

$$\mathrm{Adv}^D\Big(p_S\big(t, j-1, |P|, \ldots, p(\eta)-j-1, |q|, p_P(t), \ldots, p_Q(t)\big)\Big)$$

Thus, for all j, the advantage of any attacker against the corresponding game is uniformly bounded by:

$$\mathrm{Adv}^D\Big(p_S\big(t, p(\eta), |P|, \ldots, p(\eta), |q|, p_P(t), \ldots, p_Q(t)\big)\Big)$$

We then conclude with an hybrid argument.

□

F.5 Key Exchanges

We first prove a proposition which allows to reduce the security of n sessions in parallel to the security of one session with $N-1$ sessions in parallel. It is expressed in a more general way than required for basic key exchanges, so that we can reuse it for other results.

Proposition 39. *Let $\mathcal{O}$ be an oracle and $KE_i[\_1,\_2] := I_i(lsid_i^I, id^I);\_1 \| R_i(lsid_i^R, id^R);\_2$ a key exchange protocol, such that I binds $x^I, x_{id}^I, x_{lsid}^I$, R binds $x^R, x_{id}^R, x_{lsid}^R$ and $\mathcal{N}_l(KE)$ is disjoint of the oracle support. Let id^I, id^R be names, $\overline{s}^I = \{lsid_i^I\}_{i\in\mathbb{N}}$, $\overline{s}^R = \{lsid_i^R\}_{i\in\mathbb{N}}$, $\overline{s} = \overline{s}^I \cup \overline{s}^R$ sets of names,*

Let $T_1(\overline{x})$,$T_2(\overline{x})$, $S_1(\overline{x})$,$S_2(\overline{x})$ be parametric processes with completely disjoint names. Let N be an integer (which may depend on η), and let $\overline{s} = \{lsid_i^I, lsid_i^R\}_{1\le i\le N}$ and $\mathcal{O}_{\overline{s}}$ an oracle. If $\overline{s}$ is disjoint of the support of $\mathcal{O}$ and if,

1. *$\nu\overline{s}.\mathtt{out}(\overline{s})$ is $\mathcal{O}_{\overline{s}}$-simulatable.*

2.
$$
\begin{array}{l}
\|^{i\le N-1} KE_i[\mathtt{out}(\langle x^I, lsid^I, x_{lsid}^I, x_{id}^I\rangle), \mathtt{out}(\langle x^R, lsid^R, x_{lsid}^R, x_{id}^R\rangle)] \\
\quad \|KE_n[\ \ \mathsf{if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
\qquad\qquad S1(x^I, lsid^I, x_{lsid}^I, x_{id}^I) \\
\qquad\quad \mathsf{else}\ \mathtt{out}(\langle x^I, lsid^I, x_{lsid}^I, x_{id}^I\rangle), \\
\qquad\quad \mathsf{if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
\qquad\qquad S_2(x^R, lsid^R, x_{lsid}^R, x_{id}^R) \\
\qquad\quad \mathsf{else}\ \mathtt{out}(\langle x^R, lsid^R, x_{lsid}^R, x_{id}^R\rangle)] \\
\qquad \|^{i\le N-1} KE_i[\mathtt{out}(\langle x^I, lsid^I, x_{lsid}^I, x_{id}^I\rangle), \mathtt{out}(\langle x^R, lsid^R, x_{lsid}^R, x_{id}^R\rangle)] \\
\qquad \|\ \ KE_n[\ \ \mathsf{if}\ x_{lsid}^I = lsid_n^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
\qquad\qquad\quad \mathtt{out}(\langle k, lsid_n^I, x_{lsid}, x_{id}\rangle) \\
\qquad\qquad \mathsf{if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
\qquad\qquad\quad T_1(x^I, lsid_n^I, x_{lsid}, x_{id}) \\
\qquad\qquad \mathsf{else}\ \mathtt{out}(\langle x^I, lsid^I, x_{lsid}^I, x_{id}^I\rangle), \\
\qquad\qquad \mathsf{if}\ x_{lsid}^R = lsid_n^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
\qquad\qquad\quad \mathtt{out}(\langle k, lsid_n^R, x_{lsid}, x_{id}\rangle) \\
\qquad\qquad \mathsf{if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
\qquad\qquad\quad T_2(x^R, lsid_n^R, x_{lsid}, x_{id}) \\
\qquad\qquad \mathsf{else}\ \mathtt{out}(x^R, lsid^R, x_{lsid}^R, x_{id}^R)
\end{array}
\quad \cong_{\mathcal{O},\mathcal{O}_{\overline{s}}}
$$

Then:

$$
\begin{array}{l}
\|^{i\leq N} KE_i[\quad \textit{if } x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R \textit{ then} \\
\qquad\qquad S1(x^I, lsid^I, x^I_{lsid}, x^I_{id}) \\
\qquad\quad \textit{else } \mathtt{out}(\langle x^I, lsid^I, x^I_{lsid}, x^I_{id}\rangle), \\
\qquad\quad \textit{if } x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I \textit{ then} \\
\qquad\qquad S_2(x^R, lsid^R, x^R_{lsid}, x^R_{id}) \\
\qquad\quad \textit{else } \mathtt{out}(\langle x^R, lsid^R, x^R_{lsid}, x^R_{id}\rangle)] \\
\cong_{\mathcal{O}} \\
\qquad \|^{i\leq N}\ KE_i[\ \underset{1\leq j\leq N}{\textit{if}}\ x^I_{lsid} = lsid^R_j \wedge x^I_{id} = id_R \textit{ then} \\
\qquad\qquad\qquad \mathtt{out}(\langle k_{i,j}, lsid^I_i, x_{lsid}, x_{id}\rangle) \\
\qquad\qquad\quad \textit{if } x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R \textit{ then} \\
\qquad\qquad\qquad T_1(x^I, lsid^I_n, x_{lsid}, x_{id}) \\
\qquad\qquad\quad \textit{else } \mathtt{out}(\langle x^I, lsid^I, x^I_{lsid}, x^I_{id}\rangle), \\
\qquad\qquad\quad \underset{1\leq j\leq N}{\textit{if}}\ x^R_{lsid} = lsid^I_j \wedge x^R_{id} = id^I) \textit{ then} \\
\qquad\qquad\qquad \mathtt{out}(\langle k_{j,i}, lsid^R_i, x_{lsid}, x_{id}\rangle) \\
\qquad\qquad\quad \textit{if } x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I \textit{ then} \\
\qquad\qquad\qquad T_2(x^R, lsid^R_n, x_{lsid}, x_{id}) \\
\qquad\qquad\quad \textit{else } \mathtt{out}(\langle x^R, lsid^R, x^R_{lsid}, x^R_{id}\rangle)]
\end{array}
$$

Proof. We fix N and define an ordering (arbitrary) on the couples $(i,j)_{1\leq i,j\leq N}$. We then set:

$$
\begin{array}{l}
G^0_{(i,j)} := \\
\qquad \|_{r\leq N} KE_r[\ \underset{(r,t)\geq(i,j)}{\mathsf{if}}\ x^I_{lsid} = lsid^R_t \wedge x^I_{id} = id_R \ \mathsf{then} \\
\qquad\qquad\qquad \mathtt{out}(\langle k_{r,t}, lsid^I_r, x^I_{lsid}, x^I_{id}\rangle) \\
\qquad\qquad\quad \underset{(r,t)\geq(i,j)}{\mathsf{if}}\ x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R \ \mathsf{then} \\
\qquad\qquad\qquad T_1(x^I, lsid^I_r, x^I_{lsid}, x^I_{id}) \\
\qquad\qquad\quad \mathsf{else\ if}\ x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R \ \mathsf{then} \\
\qquad\qquad\qquad S1(x^I, lsid^I, x^I_{lsid}, x^I_{id}) \\
\qquad\qquad\quad \mathsf{else}\ \mathtt{out}(\langle x^I, lsid^I, x^I_{lsid}, x^I_{id}\rangle), \\
\\
\qquad\qquad\quad \underset{(t,r)\geq(i,j)}{\mathsf{if}}\ x^R_{lsid} = lsid^I_t \wedge x^R_{id} = id^I) \ \mathsf{then} \\
\qquad\qquad\qquad \mathtt{out}(\langle k_{t,r}, lsid^R_r, x^R_{lsid}, x^R_{id}\rangle) \\
\qquad\qquad\quad \underset{(t,r)\geq(i,j)}{\mathsf{if}}\ x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I \ \mathsf{then} \\
\qquad\qquad\qquad T_2(x^R, lsid^R_r, x^R_{lsid}, x^R_{id}) \\
\qquad\qquad\quad \mathsf{else\ if}\ x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I \ \mathsf{then} \\
\qquad\qquad\qquad S_2(x^R, lsid^R, x^R_{lsid}, x^R_{id}) \\
\qquad\qquad\quad \mathsf{else}\ \mathtt{out}(\langle x^R, lsid^R, x^R_{lsid}, x^R_{id}\rangle)]
\end{array}
$$

and

$$
\begin{aligned}
G^1_{(i,j)} := \\
\|_{r\leq N} KE_r[\ & \mathsf{if}\underset{(r,t)>(i,j)}{} x^I_{lsid} = lsid^R_t \wedge x^I_{id} = id_R \ \mathsf{then} \\
& \quad \mathtt{out}(\langle k_{r,t}, lsid^I_r, x^I_{lsid}, x^I_{id}\rangle) \\
& \mathsf{if}\underset{(r,t)>(i,j)}{} x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R \ \mathsf{then} \\
& \quad T_1(x^I, lsid^I_r, x^I_{lsid}, x^I_{id}) \\
& \mathsf{else\ if}\ x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R \ \mathsf{then} \\
& \quad S1(x^I, lsid^I, x^I_{lsid}, x^I_{id}) \\
& \mathsf{else}\ \mathtt{out}(\langle x^I, lsid^I, x^I_{lsid}, x^I_{id}\rangle), \\
\\
& \mathsf{if}\underset{(t,r)>(i,j)}{} x^R_{lsid} = lsid^I_t \wedge x^R_{id} = id^I) \ \mathsf{then} \\
& \quad \mathtt{out}(\langle k_{t,r}, lsid^R_r, x^R_{lsid}, x^R_{id}) \\
& \mathsf{if}\underset{(t,r)>(i,j)}{} x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I \ \mathsf{then} \\
& \quad T_2(x^R, lsid^R_r, x^R_{lsid}, x^R_{id}) \\
& \mathsf{else\ if}\ x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I \ \mathsf{then} \\
& \quad S_2(x^R, lsid^R, x^R_{lsid}, x^R_{id}) \\
& \mathsf{else}\ \mathtt{out}(\langle x^R, lsid^R, x^R_{lsid}, x^R_{id}\rangle)]
\end{aligned}
$$

We note that $G^1_{(i,j)} = G^0_{(i,j)+1}$, that $G^0_{(0,0)}$ is the game on the right hand side of the goal, and that $G^0_{(n,n)}$ is the game on the left hand side of the goal.

Thus, if we have uniformly that $G^1_{(i,j)} \cong G^0_{(i,j)}$, we can conclude with a classical hybrid argument.

We remark that $G^1_{(i,j)}$ and $G^0_{(i,j)}$ only differ in two places, where a conditional is added in I_i and one in R_j.

Let us fix (i,j), we define the substitution $\sigma := \{lsid^I_n \mapsto lsid^I_i, lsid^R_N \mapsto lsid^R_j, lsid^I_i \mapsto lsid^I_n, lsid^R_j \mapsto lsid^R_n\}$ and denote $\overline{s}' = \overline{s}\sigma$. We apply the substitution both to the oracle and the protocol, and the hypothesis allows us to get, for all N:

$$
\begin{array}{l}
\|^{(r,s)\neq(i,j)} I_r(lsid_r^I, id^I); \mathtt{out}(\langle x^I, lsid_r^I, x_{lsid}^I, x_{id}^I\rangle) \| R_s(lsid_s^R, id^R); \mathtt{out}(\langle x^R, lsid_s^R, x_{lsid}^R, x_{id}^R\rangle) \\
\quad \| I_i(lsid_i^I, id^I); \quad \mathsf{if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
\qquad\qquad S1(x^I, lsid_i^I, x_{lsid}^I, x_{id}^I) \\
\qquad\quad \mathsf{else}\ \mathtt{out}(\langle x^I, lsid_i^I, x_{lsid}^I, x_{id}^I\rangle), \\
\quad \| R_j(lsid_j^R, id^R)[\quad \mathsf{if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
\qquad\qquad S_2(x^R, lsid_j^R, x_{lsid}^R, x_{id}^R) \\
\qquad\quad \mathsf{else}\ \mathtt{out}(\langle x^R, lsid_j^R, x_{lsid}^R, x_{id}\rangle) \\
\cong_{\mathcal{O},\mathcal{O}_{\overline{s}'}} \\
\|^{(r,s)\neq(i,j)} I_r(lsid_r^I, id^I); \mathtt{out}(\langle x^I, lsid_r^I, x_{lsid}^I, x_{id}^I\rangle) \| R_s(lsid_s^R, id^R); \mathtt{out}(\langle x^R, lsid_s^R, x_{lsid}^R, x_{id}^R\rangle) \\
\quad \| \quad I_i(lsid_i^I, id^I); \quad \mathsf{if}\ x_{lsid}^I = lsid_j^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
\qquad\qquad \mathtt{out}(\langle k, lsid_i^I, x_{lsid}^I, x_{id}^I\rangle) \\
\qquad\qquad \mathsf{if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
\qquad\qquad\quad T_1(x^I, lsid_r^I, x_{lsid}^I, x_{id}^I) \\
\qquad\qquad \mathsf{else}\ \mathtt{out}(\langle x^I, lsid_i^I, x_{lsid}, x_{id}\rangle) \\
\quad \| \quad R_j(lsid_j^R, id^R)[\quad \mathsf{if}\ x_{lsid}^R = lsid_i^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
\qquad\qquad \mathtt{out}(\langle k, lsid_i^R, x_{lsid}^R, x_{id}^R\rangle) \\
\qquad\qquad \mathsf{if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
\qquad\qquad\quad T_2(x^R, lsid_r^R, x_{lsid}^R, x_{id}^R) \\
\qquad\qquad \mathsf{else}\ \mathtt{out}(\langle x^R, lsid_i^R, x_{lsid}^R, x_{id}^R\rangle)
\end{array}
$$

We remark that, for any r:

$$\nu\overline{s}.in(x,\overline{y}); \underset{(r,t)>(i,j)}{\mathsf{if}}\ x_{\overline{lsid}}^{\overline{=}} lsid_t^R \wedge x_{id} = id^R\ \mathsf{then}\ \mathtt{out}(k_{r,t},\overline{y})\ \mathsf{else}\ \mathtt{out}(x,\overline{y})$$

and

$$\nu\overline{s}.in(x,\overline{y}); \underset{(t,r)>(i,j)}{\mathsf{if}}\ x_{lsid}^R = lsid_t^I \wedge x_{id} = id^R\ \mathsf{then}\ \mathtt{out}(k_{t,r},\overline{y})\ \mathsf{else}\ \mathtt{out}(x,\overline{y})$$

and (resp. with S_1)

$$\nu\overline{s}.in(\overline{y}); \underset{(r,t)>(i,j)}{\mathsf{if}}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathsf{then}\ T_1(\overline{y})$$

and (resp. with S_2)

$$\nu\overline{s}.in(\overline{y}); \underset{(t,r)>(i,j)}{\mathsf{if}}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I\ \mathsf{then}\ T_2(\overline{y})$$

are $\mathcal{O}_{\overline{s}}$-simulatable by the attacker as all $lsid_j^R, lsid_j^I$ are simulatable with $\mathcal{O}_{\overline{s}}$

They are then all simulatable in parallel at the same time (Theorem 1) and using function application (Theorem 4), we get:

$$\|_{r\le N} KE_r[\ \underset{(r,t)>(i,j)}{\mathsf{if}}\ x^I_{lsid} = lsid^R_t \wedge x^I_{id} = id_R\ \mathsf{then}\ \mathtt{out}(\langle k_{r,t}, lsid^I_r, x^I_{lsid}, x^I_{id}\rangle)\ \underset{(r,t)>(i,j)}{\mathsf{if}}\ x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R\ \mathsf{then}\ T_1(x^I, lsid^I_r, x^I_{lsid}, x^I_{id})\ \mathsf{else\ if}\ x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R\ \mathsf{then}\ S1(x^I, lsid^I, x^I_{lsid}, x^I_{id})\ \mathsf{else}\ \mathtt{out}(\langle x^I, lsid^I, x^I_{lsid}, x^I_{id}\rangle),$$

$$\underset{(t,r)>(i,j)}{\mathsf{if}}\ x^R_{lsid} = lsid^I_t \wedge x^R_{id} = id^I)\ \mathsf{then}\ \mathtt{out}(\langle k_{t,r}, lsid^R_r, x^R_{lsid}, x^R_{id}\rangle)\ \underset{(t,r)>(i,j)}{\mathsf{if}}\ x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I\ \mathsf{then}\ T_2(x^R, lsid^R_r, x^R_{lsid}, x^R_{id})\ \mathsf{else\ if}\ x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I\ \mathsf{then}\ S_2(x^R, lsid^R, x^R_{lsid}, x^R_{id})\ \mathsf{else}\ \mathtt{out}(\langle x^R, lsid^R, x^R_{lsid}, x^R_{id}\rangle)]$$

$\cong_{\mathcal{O}}$

$$\|_{(r,s)\neq(i,j)} I_r(lsid^r_I, id_I);\ \underset{(r,t)>(i,j)}{\mathsf{if}}\ x^I_{lsid} = lsid^R_t \wedge x^I_{id} = id_R\ \mathsf{then}\ \mathtt{out}(\langle k_{r,t}, lsid^I_r, x^I_{lsid}, x^I_{id}\rangle)\ \underset{(r,t)>(i,j)}{\mathsf{if}}\ x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R\ \mathsf{then}\ T_1(x^I, lsid^I_r, x^I_{lsid}, x^I_{id})\ \mathsf{else\ if}\ x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R\ \mathsf{then}\ S1(x^I, lsid^I, x^I_{lsid}, x^I_{id})\ \mathsf{else}\ \mathtt{out}(\langle x^I, lsid^I, x^I_{lsid}, x^I_{id}\rangle)$$

$$\|R_s(lsid^s_R, id_R);\ \underset{(t,r)>(i,j)}{\mathsf{if}}\ x^R_{lsid} = lsid^I_t \wedge x^R_{id} = id^I)\ \mathsf{then}\ \mathtt{out}(\langle k_{t,r}, lsid^R_s, x^R_{lsid}, x^R_{id}\rangle)\ \underset{(t,r)>(i,j)}{\mathsf{if}}\ x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I\ \mathsf{then}\ T_2(x^R, lsid^R_s, x^R_{lsid}, x^R_{id})\ \mathsf{else\ if}\ x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I\ \mathsf{then}\ S_2(x^R, lsid^R_s, x^R_{lsid}, x^R_{id})\ \mathsf{else}\ \mathtt{out}(\langle x^R, lsid^R_s, x^R_{lsid}, x^R_{id}\rangle)$$

$$I_i(lsid^i_I, id_I);\ \underset{(i,t)>(i,j)}{\mathsf{if}}\ x^I_{lsid} = lsid^R_t \wedge x^I_{id} = id_R\ \mathsf{then}\ \mathtt{out}(\langle k_{r,t}, lsid^I_i, x^I_{lsid}, x^I_{id}\rangle)\ \mathsf{if}\ x^I_{lsid} = lsid^R_j \wedge x^I_{id} = id_R)\ \mathsf{then}\ \mathtt{out}(\langle k, lsid^I_i, x^I_{lsid}, x^I_{id}\rangle)\ \underset{(i,t)>(i,j)}{\mathsf{if}}\ x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R\ \mathsf{then}\ T_1(x^I, lsid^I_i, x^I_{lsid}, x^I_{id})\ \mathsf{else\ if}\ x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R\ \mathsf{then}\ S1(x^I, lsid^I_i, x^I_{lsid}, x^I_{id})\ \mathsf{else}\ \mathtt{out}(\langle x^I, lsid^I_i, x^I_{lsid}, x^I_{id}\rangle)$$

$$R_j(lsid^j_R, id_R);\ \underset{(t,j)>(i,j)}{\mathsf{if}}\ x^R_{lsid} = lsid^I_t \wedge x^R_{id} = id^I)\ \mathsf{then}\ \mathtt{out}(\langle k_{t,r}, lsid^R_j, x^R_{lsid}, x^R_{id}\rangle)\ \mathsf{if}\ x^R_{lsid} = lsid^I_i \wedge x^R_{id} = id_I)\ \mathsf{then}\ \mathtt{out}(\langle k, lsid^R_j, x^R_{lsid}, x^R_{id}\rangle)\ \underset{(t,j)>(i,j)}{\mathsf{if}}\ x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I\ \mathsf{then}\ T_2(x^R, lsid^R_j, x^R_{lsid}, x^R_{id})\ \mathsf{else\ if}\ x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I\ \mathsf{then}\ S_2(x^R, lsid^R_j, x^R_{lsid}, x^R_{id})\ \mathsf{else}\ \mathtt{out}(\langle x^R, lsid^R_j, x^R_{lsid}, x^R_{id}\rangle)$$

After α-renaming k into $k_{i,j}$, this is exactly $G^1_{(i,j)} \cong G^0_{(i,j)}$, which concludes the proof. Note that the advantage, for any (i,j), against $G^1_{(i,j)} \cong G^0_{(i,j)}$ is bounded, using the bound from Theorem 4, by the the advantage against $G^1_{(0,0)} \cong G^0_{(0,0)}$, the case where the most things are simulated.

□

Corollary 1. *Let $\mathcal{O}_{ke}$, $\mathcal{O}$ be oracles and $KE_i[\_1,\_2] := I(lsid^I_i, id^I);\_1 \| R(lsid^R_i, id^R);\_2$ a key exchange protocol, such that I binds $x^I, x^I_{id}, x^I_{lsid}$, R binds $x^R, x^R_{id}, x^R_{lsid}$ and $\mathcal{N}_l(KE)$ is disjoint of the oracle support. Let id^I, id^R be names and $\overline{s}^I = \{lsid^I_i\}_{i\in\mathbb{N}}, \overline{s}^R = \{lsid^R_i\}_{i\in\mathbb{N}}$ sets of names :*

1. $\forall i \geq 1, (\nu lsid^I_i, id^I, lsid^R_i, id^R.$

$$KE_i[\mathtt{out}(\langle x^I, lsid^I_i, x^I_{lsid}, x^I_{id}\rangle), \mathtt{out}(\langle x^R, lsid^R_i, x^R_{lsid}, x^R_{id}\rangle)] \| \mathtt{out}(\langle lsid^R_i, lsid^I_i\rangle)$$

   is $\mathcal{O}_{ke}$ simulatable)).

2. *$\overline{s}$ is disjoint of the support of $\mathcal{O}$.*

$$\begin{array}{l} KE_0[\mathtt{out}(\langle x^I, lsid^I_0, x^I_{lsid}, x^I_{id}\rangle), \mathtt{out}(\langle x^R, lsid^R_0, x^R_{lsid}, x^R_{id}\rangle)] \cong_{\mathcal{O}_{ke},\mathcal{O}} \\ KE_0[\quad \mathsf{if}\ x^I_{lsid} = lsid^R_0 \wedge x^I_{id} = id^R\ \mathsf{then} \\ \qquad\quad \mathtt{out}(\langle k, lsid^I_0, x_{lsid}, x_{id}\rangle) \\ \qquad \mathsf{else\ if}\ x^I_{lsid} \notin \overline{s}^R \wedge x^I_{id} = id^R\ \mathsf{then} \\ \qquad\quad \bot \\ \qquad \mathsf{else}\ out(\langle x^I, lsid^I_0, x^I_{lsid}, x^I_{id}\rangle), \\ \qquad \mathsf{if}\ x^R_{lsid} = lsid^I_0 \wedge x^R_{id} = id^I\ \mathsf{then} \\ \qquad\quad \mathtt{out}(\langle k, lsid^R_0, x^R_{lsid}, x^R_{id'}\rangle) \\ \qquad \mathsf{else\ if}\ x^R_{lsid} \notin \overline{s}^I \wedge x^R_{id} = id^I\ \mathsf{then} \\ \qquad\quad \bot \\ \qquad \mathsf{else}\ \mathtt{out}(\langle x^R, lsid^R_0, x^R_{lsid}, x^R_{id}\rangle)] \end{array}$$

Then, for any N which depends on the security parameter:

$$\begin{array}{l} \|^{i\leq N} KE_i[\mathtt{out}(x^I), \mathtt{out}(x^R)] \cong_{\mathcal{O}} \\ \qquad \|^{i\leq N} \quad KE_i[\quad \mathsf{if}\ (x^I_{id} = id^R)\ \mathsf{then} \\ \qquad\qquad\qquad \underset{1\leq j\leq N}{\mathsf{if}}\ x^I_{lsid} = lsid^R_j \wedge x^I_{id} = id_R\ \mathsf{then} \\ \qquad\qquad\qquad\qquad \mathtt{out}(k_{i,j}) \\ \qquad\qquad\qquad \mathsf{else}\ \mathtt{out}(x^I), \\ \qquad\qquad \mathsf{if}\ (x^R_{id} = id_I)\ \mathsf{then} \\ \qquad\qquad\qquad \underset{1\leq j\leq N}{\mathsf{if}}\ x^R_{lsid} = lsid^I_j \wedge x^R_{id} = id_I\ \mathsf{then} \\ \qquad\qquad\qquad\qquad \mathtt{out}(k_{j,i}) \\ \qquad\qquad\qquad \mathsf{else}\ \mathtt{out}(x^R)] \end{array}$$

Proof. Let us fix N, which may depend on the security parameter.

Ry direct application of Theorem 5, with $P := I(lsid^I, id^I); \mathtt{out}(\langle x^I, lsid^I, x^I_{lsid}, x^I_{id}\rangle) \| R(lsid^R, id^R); \mathtt{out}(\langle x^R, l$

$R := KE$, and Q being the right handside of hypothesis (3), we get that:

$$
\begin{array}{l}
\|^{i\leq N} KE_i[\mathtt{out}(\langle x^I, lsid_i^I, x_{lsid}^I, x_{id}^I\rangle), \mathtt{out}(\langle x^R, lsid_i^R, x_{lsid}^R, x_{id}^R\rangle)] \\
\cong_{\mathcal{O},\mathcal{O}_{ke}} \\
\quad \|^{i\leq N-1} KE_i[\mathtt{out}(\langle x^I, lsid_i^I, x_{lsid}^I, x_{id}^I\rangle), \mathtt{out}(\langle x^R, lsid_i^R, x_{lsid}^R, x_{id}^R\rangle)] \\
\quad \|KE_0[\ \ \mathsf{if}\ x_{lsid}^I = lsid^R \wedge x_{id} = id^R\ \mathsf{then} \\
\qquad\qquad \mathtt{out}(\langle k, lsid^I, x_{lsid}, x_{id}\rangle) \\
\qquad\quad \mathsf{else\ if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
\qquad\qquad \bot \\
\qquad\quad \mathsf{else}\ out(\langle x^I, lsid^I, x_{lsid}^I, x_{id}^I\rangle), \\
\qquad\quad \mathsf{if}\ x_{lsid}^R = lsid^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
\qquad\qquad \mathtt{out}(\langle k, lsid^R, x_{lsid}^R, x_{id}^R\rangle) \\
\qquad\quad \mathsf{else\ if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
\qquad\qquad \bot \\
\qquad\quad \mathsf{else}\ \mathtt{out}(\langle x^R, lsid^R, x_{lsid}^R, x_{id}^R\rangle)]
\end{array}
$$

This allows us to obtain the hypothesis of Proposition 39, where $\mathcal{O}_{\overline{s}}$ is instantiated with $\mathcal{O}_{ke}$. We thus conclude using Proposition 39.

□

Corollary 2. *Let $\mathcal{O}_T$, $\mathcal{O}_{ke}$, $\mathcal{O}_r$,$\mathcal{O}_{P,Q}$ be oracles and*

$KE_i[\__1, \__2] := I(lsid_i^I, id^I);\__1 \| R(lsid_i^R, id^R);\__2$ a key exchange protocol, such that I binds $x^I, x_{id}^I, x_{lsid}^I$, R binds $x^R, x_{id}^R, x_{lsid}^R$ and $\mathcal{N}_l(KE)$ is disjoint of the oracle support. Let id^I, id^R be names, $\overline{s}^I = \{lsid_i^I\}_{i\in\mathbb{N}}$,$\overline{s}^R = \{lsid_i^R\}_{i\in\mathbb{N}}$ and $\overline{s} = \overline{s}^I \cap \overline{s}^R$ sets of names.

Let $\overline{p} = \{id^I, id^R\}$, $P(x, \overline{y}) = P_1(x, \overline{y}) \| P_2(x, \overline{y})$ and $Q(x, \overline{y}, \overline{z}) = Q_1(x, \overline{y}, \overline{z}) \| Q_1(x, \overline{y}, \overline{z})$ be parameterized protocols, such that $\mathcal{N}_l(P, Q)$ is disjoint of the oracle support.

I-1 $\forall i \geq 1, (\nu lsid_i^I, id^I, lsid_i^R, id^R.KE_i[\mathtt{out}(x^I), \mathtt{out}(x^R)] \| \mathtt{out}(\langle lsid_i^R, lsid_i^I\rangle)$ is $\mathcal{O}_T$-simulatable)).

I-2 $\overline{s}$ is disjoint of the support of $\mathcal{O}_{P,Q}$.

$$
\begin{array}{ll}
& KE_0[\mathtt{out}(\langle x^I, lsid_0^I, x_{lsid}^I, x_{id}^I\rangle), \mathtt{out}(\langle x^R, lsid_0^R, x_{lsid}^R, x_{id}^R\rangle) \cong_{\mathcal{O}_T,\mathcal{O}_{P,Q}} \\
KE_0 & [\mathit{if}\ x_{lsid}^I = lsid_0^R \wedge x_{id}^I = id^R\ \mathit{then} \\
& \quad \mathtt{out}(\langle k, lsid_0^I, x_{lsid}^I, x_{id}^I\rangle) \\
& \quad \mathit{else\ if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathit{then} \\
& \quad \bot \\
\text{I-3} & \quad \mathit{else}\ \mathtt{out}(\langle x^I, lsid_0^I, x_{lsid}^I, x_{id}^I\rangle), \\
& \quad \mathit{if}\ x_{lsid}^R = lsid^I \wedge x_{id}^R = id^I\ \mathit{then} \\
& \quad \mathtt{out}(\langle k, lsid_0^R, x_{lsid}^R, x_{id}^R\rangle) \\
& \quad \mathit{else\ if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I\ \mathit{then} \\
& \quad \bot \\
& \quad \mathit{else}\ \mathtt{out}(\langle x^R, lsid_0^R, x_{lsid}^R, x_{id}^R\rangle)]
\end{array}
$$

and

R-1 $\forall\ 1 \leq i, j \leq n, \nu\overline{p}, k_{i,j}.P_0(\overline{p}, k_{i,j})$ is $\mathcal{O}_r$-simulatable.

R-2 $\forall\ 1 \leq i \leq n, \nu\overline{p}, k_{i,j}.Q_0(\overline{p}, k_{i,j})$ is $\mathcal{O}_r$-simulatable.

R-3 $\overline{s}$ is disjoint of the support of $\mathcal{O}_k$.

R-4 $P_0(\overline{p},k) \cong_{\mathcal{O}_r,\mathcal{O}_{ke}} Q_0(\overline{p},k)$

and

C-1 $\nu\overline{p}.in(x_i^I).P_i^I(x_i^I) \| in(x_i^R).P_i^R(x_i^R)$ *is* $\mathcal{O}_{P,Q}$*-simulatable.*

1. $\nu\overline{p}.$
$$\begin{array}{l} \|^{i\leq n} \quad KE_i[\quad \mathit{if}\,(x_{id}^I = id^R)\ \mathit{then} \\ \qquad \underset{1\leq j\leq n}{\mathit{if}}\,(x_{lsid}^I = lsid_j^R \wedge x_{id}^I = id_R)\ \mathit{then} \\ \qquad\qquad out(\langle i,j\rangle) \\ \qquad \mathit{else}\ P_i^I(x_i^I), \\ \qquad \mathit{if}\,(x_{id}^R = id_I)\ \mathit{then} \\ \qquad \underset{1\leq j\leq n}{\mathit{if}}\,(x_{lsid}^R = lsid_j^I \wedge x_{id}^R = id_I)\ \mathit{then} \\ \qquad\qquad out(\langle i,j\langle) \\ \qquad \mathit{else}\ P_i^R(x_i^R)] \end{array} \qquad \textit{is } \mathcal{O}_{ke}\textit{-simulatable.}$$

Then, for any n which may depend on the security parameter:

$$\|^{i\leq n} KE_i[P_i^I(x_i^I), P_i^R(x_i^R)] \cong$$
$$\|^{i\leq n} KE_i[\mathit{if}\, x_{id}^I = id^R\ \mathit{then}\ Q_i^I(x_i^I)\ \mathit{else}\ P_i^I(x_i^I), \mathit{if}\, x_{id}^R = id^I\ \mathit{then}\ Q_i^R(x_i^R)\ \mathit{else}\ P_i^R(x_i^R)]$$

Proof. Using Corollary 1 on hypothesis A-1,A-2 and A-3, we get that, for all N:

$$\begin{array}{l} \|^{i\leq N} KE_i[\mathtt{out}(x^I), \mathtt{out}(x^R)] \cong_{\mathcal{O}} \\ \quad \|^{i\leq N} \quad KE_i[\quad \mathsf{if}\ (x_{id}^I = id^R)\ \mathsf{then} \\ \qquad \underset{1\leq j\leq N}{\mathsf{if}}\, x_{lsid}^I = lsid_j^R \wedge x_{id}^I = id_R\ \mathsf{then} \\ \qquad\qquad \mathtt{out}(k_{i,j}) \\ \qquad \mathsf{else}\ \mathtt{out}(x^I), \\ \qquad \mathsf{if}\ (x_{id}^R = id_I)\ \mathsf{then} \\ \qquad \underset{1\leq j\leq N}{\mathsf{if}}\, x_{lsid}^R = lsid_j^I \wedge x_{id}^R = id_I\ \mathsf{then} \\ \qquad\qquad \mathtt{out}(k_{j,i}) \\ \qquad \mathsf{else}\ \mathtt{out}(x^R)] \end{array}$$

Now, as $\nu\overline{p}, lsid_i^I, lsid_i^R.in(x).P(x) \| in(x).Q(x)$ is $\mathcal{O}_p$-simulatable (hypothesis C-1), using twice Theorem 4 we get that :

$$\begin{array}{l} \|^{i\leq N} KE_i[P^I(x^I), P^R(x^R)] \cong_{\mathcal{O}_p} \\ \quad \|^{i\leq N} \quad KE_i[\quad \mathsf{if}\ (x_{id}^I = id^R)\ \mathsf{then} \\ \qquad \underset{1\leq j\leq N}{\mathsf{if}}\, x_{lsid}^I = lsid_j^R \wedge x_{id}^I = id_R\ \mathsf{then} \\ \qquad\qquad P^I(k_{i,j}) \\ \qquad \mathsf{else}\ P^I(x^I), \\ \qquad \mathsf{if}\ (x_{id}^R = id_I)\ \mathsf{then} \\ \qquad \underset{1\leq j\leq N}{\mathsf{if}}\, x_{lsid}^R = lsid_j^I \wedge x_{id}^R = id_I\ \mathsf{then} \\ \qquad\qquad P^R(k_{j,i}) \\ \qquad \mathsf{else}\ P^R(x^R)] \end{array}$$

and

$$\|^{i\leq N} KE_i[\mathsf{if}\ x^I_{id} = id^R\ \mathsf{then}\ Q^I(x^I)\ \mathsf{else}\ P^I(x^I), \mathsf{if}\ x^R_{id} = id^I\ \mathsf{then}\ Q^R(x^R)\ \mathsf{else}\ P^R(x^R)] \cong_{\mathcal{O}_p}$$

$$\begin{array}{rl}
\|^{i\leq N} & KE_i[\ \ \mathsf{if}\ (x^I_{id} = id^R)\ \mathsf{then} \\
& \quad \underset{1\leq j\leq N}{\mathsf{if}}\ x^I_{lsid} = lsid^R_j \wedge x^I_{id} = id_R\ \mathsf{then} \\
& \quad\quad Q^I(k_{i,j}) \\
& \quad \mathsf{else}\ P^I(x^I), \\
& \quad \mathsf{if}\ (x^R_{id} = id_I)\ \mathsf{then} \\
& \quad\quad \underset{1\leq j\leq N}{\mathsf{if}}\ x^R_{lsid} = lsid^I_j \wedge x^R_{id} = id_I\ \mathsf{then} \\
& \quad\quad\quad Q^R(k_{j,i}) \\
& \quad\quad \mathsf{else}\ P^R(x^R)]
\end{array}$$

Moreover, using Theorem 5 on hypothesis B-1,B-2,B-3 and B-4, we get that

$$\forall n\ \|^{i\leq N^2} P_i(\overline{p}, k_i) \cong_{\mathcal{O}_k} Q_i(\overline{p}, k_i)$$

Combined with Theorem 2 on the $\mathcal{O}_k$ simulatability of the key exchange (hypothesis C-2) we get:

$$\begin{array}{rl}
\|^{i\leq N} & KE_i[\ \ \mathsf{if}\ (x^I_{id} = id^R)\ \mathsf{then} \\
& \quad \underset{1\leq j\leq N}{\mathsf{if}}\ x^I_{lsid} = lsid^R_j \wedge x^I_{id} = id_R\ \mathsf{then} \\
& \quad\quad P^I(k_{i,j}) \\
& \quad \mathsf{else}\ P^I(x^I), \\
& \quad \mathsf{if}\ (x^R_{id} = id_I)\ \mathsf{then} \\
& \quad\quad \underset{1\leq j\leq N}{\mathsf{if}}\ x^R_{lsid} = lsid^I_j \wedge x^R_{id} = id_I\ \mathsf{then} \\
& \quad\quad\quad P^R(k_{j,i}) \\
& \quad\quad \mathsf{else}\ P^R(x^R)] \\
& \quad\quad\quad \cong \\
\|^{i\leq N} & KE_i[\ \ \mathsf{if}\ (x^I_{id} = id^R)\ \mathsf{then} \\
& \quad \underset{1\leq j\leq N}{\mathsf{if}}\ x^I_{lsid} = lsid^R_j \wedge x^I_{id} = id_R\ \mathsf{then} \\
& \quad\quad Q^I(k_{i,j}) \\
& \quad \mathsf{else}\ P^I(x^I), \\
& \quad \mathsf{if}\ (x^R_{id} = id_I)\ \mathsf{then} \\
& \quad\quad \underset{1\leq j\leq N}{\mathsf{if}}\ x^R_{lsid} = lsid^I_j \wedge x^R_{id} = id_I\ \mathsf{then} \\
& \quad\quad\quad Q^R(k_{j,i}) \\
& \quad\quad \mathsf{else}\ P^R(x^R)]
\end{array}$$

We thus conclude with transitivity.

□

Corollary 3. *Let $\mathcal{O}_{KE}$, $\mathcal{O}_r$,$\mathcal{O}_{P,Q}$ be oracles and*

$$KE_i[\__1, \__2] := I_i(lsid^I_i, id^I); \__1 | R_i(lsid^R i, id^R); \__2$$

a key exchange protocol with $I_i(lsid^I_i, id^I) := I^0_i(lsid^I_i, id^I); I^1_i(x^I)$ and $R_i(lsid^R_i, id^R) := R^0_i(lsid^R_i, id^R); R^1_i(x^R)$ such that I^0 binds x^I, x_{id}, x_{lsid}, R^0 binds x^R, x_{id}, x_{lsid} and $\mathcal{N}_l(KE)$ is disjoint of the oracles support. Let $\overline{p} = \{id^I, id^R\}$, $P_i(x, \overline{y}) = P^I_i(x, \overline{y}) \| P^R_i(x, \overline{y})$,$Q(x, \overline{y}, \overline{z}) =$

$Q_i^I(x,\overline{y},\overline{z})\|Q_i^R(x,\overline{y},\overline{z})$, $C_i(\overline{z})$ *and* $D_i(\overline{z})$ *be protocols, such that* $\mathcal{N}_l(P,Q,C,D)$ *is disjoint of the oracles support.*

Let id^I, id^R *be names,* $\overline{s}^I = \{lsid_i^I\}_{i\in\mathbb{N}}, \overline{s}^R = \{lsid_i^R\}_{i\in\mathbb{N}}$ *and* $\overline{s} = \overline{s}^I \cap \overline{s}^R$ *sets of names.*

A-1 $\forall i \in \mathbb{N}, (\nu lsid_i^I, id^I, lsid_i^R, id^R.C_i(\overline{p})\|I_i^0(lsid_i^I, id^I); \mathtt{out}(x^I)\|R_i^0(lsid_i^R, id^R); \mathtt{out}(x^R)$ *is* $\mathcal{O}_{KE}$ *simulatable)).*

A-2 $\overline{s}$ *is disjoint of the support of* $\mathcal{O}_p$.

A-3

$$
\begin{array}{ll}
C_i(\overline{p})\|I_0^0(lsid_0^I, id^I); & \textit{if } x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R \textit{ then} \\
 & \quad I^1(x^I); \mathtt{out}(x^I) \\
 & \textit{else } \mathtt{out}(\langle x^I, lsid_0^I, x_{lsid}^I, x_{id}^I\rangle) \\
\|R^0(lsid_0^R, id^R); & \textit{if } x_{lsid}^R \notin \overline{s}^I \wedge x_{id} = id^I \textit{ then} \\
 & \quad R^1(x^R); \mathtt{out}(x^R) \\
 & \textit{else } \mathtt{out}(\langle x^R, lsid^R, x_{lsid}^R, x_{id}^R\rangle) \\
\cong_{\mathcal{O}_{KE},\mathcal{O}_p} & \\
C_i(\overline{p})\|I^0(lsid_0^I, id^I); & \textit{if } x_{lsid}^I = lsid^R \wedge x_{id} = id^R \textit{ then} \\
 & \quad \mathtt{out}(\langle k, lsid_0^I, x_{lsid}^I, x_{id}^I\rangle) \\
 & \textit{else if } x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R \textit{ then} \\
 & \quad I^1(x^R); \perp \\
 & \textit{else } \mathtt{out}(\langle x^I, lsid^I, x_{lsid}^I, x_{id}^I\rangle) \\
\|R^0(lsid_0^R, id^R); & \textit{if } x_{lsid}^R = lsid^I \wedge x_{id}^R = id^I \textit{ then} \\
 & \quad \mathtt{out}(\langle k, lsid_0^R, x_{lsid}^R, x_{id}^R\rangle) \\
 & \textit{else if } x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I \textit{ then} \\
 & \quad I^1(x^R); \perp \\
 & \textit{else } \mathtt{out}(\langle x^R, lsid_0^R, x_{lsid}^R, x_{id}^R\rangle)
\end{array}
$$

and for any N *which may depend on the security parameter:*

B-1 $\|^{i\le N^2} D_i(\overline{p})\|I_i^1(k_i); P_i^I(\overline{p},k_i)\|B_i^1(k_i); P_i^R(\overline{p},k_i) \cong_{\mathcal{O}_r,\mathcal{O}_k} \|^{i\le n^2} D_i(\overline{p})\|I_i^1(k_i); Q_i^I(\overline{p},k_i)\|B_i^1(k_i); Q_i^R(\overline{p},k_i)$

and

C-1 $\nu\overline{p}, lsid_i^I, lsid_i^R.D_i(\overline{p})\|in(x).P_i(x) \ \|in(x).Q_i(x)\|in(x).I_i^1(x); P_i^I(x)\|in(x).R_i^1(x); P_i^R(x)$ $\|in(x).I_i^1(x); Q_i^I(x)\|in(x).R_i^1(x); Q_i^R(x)$ *is* $\mathcal{O}_p$ *simulatable.*

C-2 $\nu\overline{p}.$

$$
\begin{array}{lll}
\|^{i\le N} & C_i(\overline{p})\|I_i^0(lsid_i^I, id^I); & \underset{1\le j\le N}{\textit{if}} x_{lsid}^I = lsid_j^R \wedge x_{id}^I = id^R \textit{ then} \\
 & & \quad \mathtt{out}(\langle i,j\rangle) \\
 & & \textit{else if } x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R \textit{ then} \\
 & & \quad I_i^1(x^I); \perp \\
 & & \textit{else } I_i^1(x^I); P_i^I(x^I) \\
 \|R_i^0(lsid_i^R, id^R)[& & \underset{1\le j\le N}{\textit{if}} x_{lsid}^R = lsid_j^I \wedge x_{id}^R = id^I \textit{ then} \\
 & & \quad \mathtt{out}(\langle i,j\rangle) \\
 & & \textit{else if } (x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I \textit{ then} \\
 & & \quad R_i^1(x^R); \perp \\
 & & \textit{else } R_i^1(x^R); P_i^R(x^R)
\end{array}
$$

is $\mathcal{O}_k$ *simulatable.*

Then, for any n:

$$\|^{i\leq N} C_i(\overline{p})\|D_i(\overline{p})\|KE_i[P_i^I(x^I), P_i^R(x^R)] \cong$$
$$\|^{i\leq N} C_i(\overline{p})\|D_i(\overline{p})\|KE_i[\mathsf{if}\ x_{id}^I = id^R\ \mathsf{then}\ Q_i^I(x^I)\ \mathsf{else}\ P_i^I(x^I), \mathsf{if}\ x_{id}^R = id^I\ \mathsf{then}\ Q_i^R(x^R)\ \mathsf{else}\ P_i^R(x^R)]$$

Proof. Let N an integer, which may depend on the security parameter. Ry application of Theorem 5, with P and R as the left handside of hypothesis A-3, and Q being the right handside of hypothesis A-3, we get that:

$$\begin{array}{l}
\|^{i\leq n-1} C_i(\overline{p})\|I_i^0(lsid_i^I, id^I); \mathtt{out}(x^I)\|R_i^0(lsid_i^R, id^R); \mathtt{out}(x^R) \\
\quad C_n(\overline{p})\|I_n^0(lsid_n^I, id^I); \quad \mathsf{if}\ (x_{lsid}^I \notin \overline{s}^R \wedge x_{id} = id^R\ \mathsf{then} \\
\qquad\qquad\qquad\qquad I_n^1(x^I); \mathtt{out}(x^I) \\
\qquad\qquad\qquad\qquad \mathsf{else}\ \mathtt{out}(x^I) \\
\| \quad \|R_n^0(lsid_n^R, id^R); \qquad \mathsf{if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^I\ \mathsf{then} \\
\qquad\qquad\qquad\qquad R_n^1(x^R); \mathtt{out}(x^R) \\
\qquad\qquad\qquad\qquad \mathsf{else}\ \mathtt{out}(x^R) \\
\cong_{\mathcal{O}_P} \\
\quad \|^{i\leq N-1} C_i(\overline{p})\|I_i^0(lsid_i^I, id^I); \mathtt{out}(x^I)\|R_i^0(lsid_i^R, id^R); \mathtt{out}(x^R) \\
\quad \| \quad C_n(\overline{p})\|I_n^0(lsid_n^I, id^I); \quad \mathsf{if}\ x_{lsid} = lsid_R^n \wedge x_{id}^I = id^R\ \mathsf{then} \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathtt{out}(k) \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{else\ if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad I_n^1(x^I); bad \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{else}\ \mathtt{out}(x^I) \\
\qquad\quad \|R_n^0(lsid_n^R, id^R)[\qquad \mathsf{if}\ x_{lsid}^R = lsid_I^n \wedge x_{id} = id^I\ \mathsf{then} \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathtt{out}(k) \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{else\ if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad R_n^1(x^R); bad \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{else}\ \mathtt{out}(x^R)
\end{array}$$

Using Proposition 39, with $S_1 = I^1(x^I); \mathtt{out}(x^I), S_2 = R^1(x^R); \mathtt{out}(x^R), R_1 = I^1(x^I); \perp, R_2 = R^1(x^R); \perp$, we get that:

$$
\begin{array}{l}
C_i(\overline{p})\|I_i^0(lsid_i^I, id^I); \quad \mathsf{if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad I_i^1(x^I); \mathtt{out}(x^I) \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathsf{else}\ \mathtt{out}(x^I) \\
\|^{i\leq N} \quad \|R_i^0(lsid_i^R, id^R); \qquad \mathsf{if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad R_i^1(x^R); \mathtt{out}(x^R) \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathsf{else}\ \mathtt{out}(x^R)
\end{array}
$$

$\cong_{\mathcal{O}_P}$

$$
\begin{array}{lll}
\|^{i\leq N} & C_i(\overline{p})\|I_i^0(lsid_i^I, id^I); & \underset{1\leq j\leq N}{\mathsf{if}}\ x_{lsid}^I = lsid_R^j \wedge x_{id}^I = id^R\ \mathsf{then} \\
& & \quad \mathtt{out}(k_{i,j}) \\
& & \mathsf{else\ if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
& & \quad I_i^1(x^I); bad \\
& & \mathsf{else}\ \mathtt{out}(x^I) \\
& \|R_i^0(lsid_i^R, id^R)[& \underset{1\leq i\leq N}{\mathsf{if}}\ x_{lsid}^R = lsid_I^j \wedge x_{id}^R = id^I\ \mathsf{then} \\
& & \quad \mathtt{out}(k_{j,i}) \\
& & \mathsf{else\ if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
& & \quad R_i^1(x^R); bad \\
& & \mathsf{else}\ \mathtt{out}(x^R)
\end{array}
$$

Now, with this context, using twice using Theorem 4 with the simulatability of $\nu\overline{p}, lsid_i^I, lsid_i^R.D_i(\overline{p})$ $\|in(x).P_i(x)\|in(x).I_i^1(x); P_i^I(x)\|in(x).R_i^1(x); P_i^R(x)$ from C-1, we may get that:

$$
\begin{array}{l}
\|^{i\leq N} C_i(\overline{p})\|D_i(\overline{p})\|I_i^0(lsid_i^I, id^I); \quad \mathsf{if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad I_i^1(x^I); P_i^I(x^I) \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathsf{else}\ I_i^1(x^I); P_i^I(x^I) \\
\|R_i^0(lsid_i^R, id^R); \qquad\qquad\qquad \mathsf{if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id} = id^I\ \mathsf{then} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad R_i^1(x^R); P_i^R(x^R) \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathsf{else}\ R_i^1(x^R); P_i^R(x^R)
\end{array}
$$

$\cong_{\mathcal{O}_P}$

$$
\begin{array}{lll}
\|^{i\leq N} & C_i(\overline{p})\|D_i(\overline{p})\|I_i^0(lsid_i^I, id^I); & \underset{1\leq j\leq N}{\mathsf{if}}\ x_{lsid}^I = lsid_j^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
& & \quad I_i^1(k_{i,j}); P_i^1(k_{i,j}) \\
& & \mathsf{else\ if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R\ \mathsf{then} \\
& & \quad I_i^1(x^I); \perp \\
& & \mathsf{else}\ I_i^1(x^I); P_i^I(x^I) \\
& \|R_i^0(lsid_i^R, id^R)[& \underset{1\leq k\leq m}{\mathsf{if}}\ x_{lsid}^R = lsid_j^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
& & \quad R_i^1(k_{j,i}); P_i^R(k_{j,i}) \\
& & \mathsf{else\ if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I\ \mathsf{then} \\
& & \quad R_i^1(x^R); \perp \\
& & \mathsf{else}\ R_i^1(x^R); P_i^R(x^R)
\end{array}
$$

We can simplify the left handside of the equivalence and get that:

$$
\begin{array}{ll}
\|^{i\leq N} C_i(\overline{p})\|D_i(\overline{p})\|I_i^0(lsid_i^I, id^I); & I_i^1(x^I); P_i^I(x^I) \\
\|R_i^0(lsid_i^R, id^R); & R_i^1(x^R); P_i^R(x^R)
\end{array}
$$

$\cong_{\mathcal{O}_P}$

$$
\begin{array}{lll}
\|^{i\leq N} & C_i(\overline{p})\|D_i(\overline{p})\|I_i^0(lsid_i^I, id^I); & \mathsf{if} \underset{1\leq j\leq m}{} x_{lsid}^I = lsid_R^j \wedge x_{id} = id^R \ \mathsf{then} \\
& & \quad I_i^1(k_{i,j}); P_i^1(k_{i,j}) \\
& & \mathsf{else\ if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R \ \mathsf{then} \\
& & \quad I_i^1(x^I); bad \\
& & \mathsf{else}\ I_i^1(x^I); P_i^I(x^I) \\
& \|R_i^0(lsid_i^R, id^R); & \mathsf{if} \underset{1\leq j\leq N}{} x_{lsid}^R = lsid_I^j \wedge x_{id}^R = id^I \ \mathsf{then} \\
& & \quad R_i^1(k_{j,i}); P_i^R(k_{j,i}) \\
& & \mathsf{else\ if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I \ \mathsf{then} \\
& & \quad R_i^1(x^R); \bot \\
& & \mathsf{else}\ R_i^1(x^R); P_i^R(x^R)
\end{array}
$$

Ry performing the same operation with Q, we can also get:

$$
\begin{array}{lll}
 & C_i(\overline{p})\|D_i(\overline{p})\|I_i^0(lsid_i^I, id^I); & \mathsf{if}\ x_{id}^I = id^R \ \mathsf{then} \\
 & & \quad I_i^1(x^I); Q_i^I(x^I) \\
 & & \mathsf{else}\ I_i^1(x^I); P_i^I(x^I) \\
\|^{i\leq N} & \|R_i^0(lsid_i^R, id^R); & \mathsf{if}\ x_{id}^R = id^I \ \mathsf{then} \\
 & & \quad R_i^1(x^R); Q_i^R(x^R) \\
 & & \mathsf{else}\ R_i^1(x^R); P_i^R(x^R)
\end{array}
$$

$\cong_{\mathcal{O}_P}$

$$
\begin{array}{lll}
\|^{i\leq N} & C_i(\overline{p})\|D_i(\overline{p})\|I_i^0(lsid_i^I, id^I); & \mathsf{if} \underset{1\leq j\leq N}{} x_{lsid}^I = lsid_j^R \wedge x_{id}^I = id^R \ \mathsf{then} \\
& & \quad I_i^1(k_{i,j}); Q_i^1(k_{i,j}) \\
& & \mathsf{else\ if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R \ \mathsf{then} \\
& & \quad I_i^1(x^I); \bot \\
& & \mathsf{else}\ I_i^1(x^I); P^I(x^I) \\
& \|R_i^0(lsid_i^R, id^R)[& \mathsf{if} \underset{1\leq j\leq N}{} x_{lsid}^R = lsid_j^I \wedge x_{id}^R = id^I \ \mathsf{then} \\
& & \quad R_i^1(k_{j,i}); Q_i^R(k_{j,i}) \\
& & \mathsf{else\ if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I \ \mathsf{then} \\
& & \quad R_i^1(x^R); \bot \\
& & \mathsf{else}\ R_i^1(x^R); P_i^R(x^R)
\end{array}
$$

To conclude with transitivity, we must prove the equivalence between the two idealized version with either P or Q.

Combining Hypothesis B-1 with Theorem 2 on the $\mathcal{O}_k$ simulatability of the key exchange (hypothesis C-2) we do get the necessary equivalence to conclude:

$$\|^{i\leq N} \quad C_i(\overline{p})\|D_i(\overline{p})\|I_i^0(lsid_i^I, id^I); \quad \begin{array}{l} \mathsf{if} \underset{1\leq j\leq N}{} x_{lsid}^I = lsid_j^R \wedge x_{id} = id^R \ \mathsf{then} \\ \quad I_i^1(k_{i,j}); P_i^1(k_{i,j}) \\ \mathsf{else\ if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R \ \mathsf{then} \\ \quad I_i^1(x^I); bad \\ \mathsf{else}\ I_i^1(x^I); P_i^I(x^I) \end{array}$$

$$\|R_i^0(lsid_i^R, id^R)[\quad \begin{array}{l} \mathsf{if} \underset{1\leq j\leq N}{} x_{lsid}^R = lsid_I^j \wedge x_{id}^R = id^I \ \mathsf{then} \\ \quad R_i^1(k_{j,i}); P_i^R(k_{j,i}) \\ \mathsf{else\ if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I \ \mathsf{then} \\ \quad R_i^1(x^R); bad \\ \mathsf{else}\ R_i^1(x^R); P_i^R(x^R) \end{array}$$

$\cong$

$$\|^{i\leq N} \quad C_i(\overline{p})\|D_i(\overline{p})\|I_i^0(lsid_i^I, id^I); \quad \begin{array}{l} \mathsf{if} \underset{1\leq j\leq N}{} x_{lsid}^I = lsid_j^R \wedge x_{id}^I = id^R \ \mathsf{then} \\ \quad I_i^1(k_{i,j}); Q_i^1(k_{i,j}) \\ \mathsf{else\ if}\ x_{lsid}^I \notin \overline{s}^R \wedge x_{id}^I = id^R \ \mathsf{then} \\ \quad I_i^1(x^I); \bot \\ \mathsf{else}\ I_i^1(x^I); P_i^I(x^I) \end{array}$$

$$\|R_i^0(lsid_i^R, id^R); \quad \begin{array}{l} \mathsf{if} \underset{1\leq j\leq N}{} x_{lsid}^R = lsid_j^I \wedge x_{id}^R = id^I \ \mathsf{then} \\ \quad R_i^1(k_{j,i}); Q_i^R(k_{j,i}) \\ \mathsf{else\ if}\ x_{lsid}^R \notin \overline{s}^I \wedge x_{id}^R = id^I \ \mathsf{then} \\ \quad R_i^1(x^R); \bot \\ \mathsf{else}\ R_i^1(x^R); P_i^R(x^R) \end{array}$$

□

F.6 Computational soundness

Lemma 4. *For protocols P, Q, A, B, an oracle $\mathcal{O}$, and a list $\mathcal{O}_l$ of protocol oracles,*

$$\mathcal{A}^{\mathcal{O},\mathcal{O}_{(A\|P)?(B\|Q)}} \prec \epsilon \Leftrightarrow \mathcal{A}^{\mathcal{O},\mathcal{O}_l,\mathcal{O}_{A?B},\mathcal{O}_{P?Q}} \prec \epsilon$$

Proof. For protocols P, Q such that $\mathcal{C}(P) \cap \mathcal{C}(Q) = \emptyset$, for any message m, random tape ρ_s and history tape θ, we have by definition of the semantic of $\|$ and the definition of the parallel oracles:

$$\mathcal{O}_{P\|Q}(\rho_s, \theta)(m) =< \mathcal{O}_P, \mathcal{O}_Q > (\rho_s, \theta)(m)$$

The desired result then immediately follows. □

Lemma 29. *Given two protocols P, Q, random tapes ρ_r, ρ_s, a cryptographic library $\mathcal{M}_f$ and an oracle $\mathcal{O}$, we have:*

$$\forall \mathcal{M} \supset \mathcal{M}_f. \quad \mathcal{M} \models_{\mathcal{O}} \widetilde{t_P} \sim \widetilde{t_Q}$$

$$\Leftrightarrow$$

$$P \cong_{\mathcal{O}} Q$$

Proof. Let us write $\widetilde{t_P} = \widetilde{t_P^0}, \ldots, \widetilde{t_P^n}$. Without loss of generality, we assume that every distinguisher makes exactly n calls to the oracle, if it is not the case we simply add dummy calls for the remaining ones.

We start by proving the top to bottom implication. Given a distinguisher $\mathcal{B}^{\mathcal{O},\mathcal{O}_{P?Q}}$ and η, ρ_r, we let $m_0, \ldots, m_k$ (resp. $m'_0, \ldots, m'_k$) be the successive contents of the oracle input tape along the computation of $\mathcal{A}^{\mathcal{O}(\rho_s,\rho_\mathcal{O}),\mathcal{O}_P(\rho_s)}$ (resp. $\mathcal{A}^{\mathcal{O}(\rho_s,\rho_\mathcal{O}),\mathcal{O}_Q(\rho_s)}$). Let $\sigma = \{x_0 \mapsto m_0, \ldots, x_k \mapsto m_k\}$ (resp. $\sigma' = \{x_1 \mapsto m'_1, \ldots, x_k \mapsto m'_k\}$). Consider now the PPTOM $\mathcal{A}^{\mathcal{O}}_{g_k}$, which, on input $b_0, \ldots, b_k, \eta, \rho_r$, executes the same code as $\mathcal{B}$, however replacing the ith call to the oracle $\mathcal{O}_P$ (resp. $\mathcal{O}_Q$), $i \leq k$, using b_i instead of the oracle reply.

The result of $\mathcal{A}^{\mathcal{O}}_{g_k}$ is then what $\mathcal{B}$ would have queried at the $k+1$ oracle call. It follows that $\mathcal{A}^{\mathcal{O}}_{g_k}(b_0, \ldots, b_k, \eta, \rho_r) = m_k$ (resp. m'_k) if b_i is the reply of $\mathcal{O}_P$ (resp. $\mathcal{O}_Q$) on the query m_i, for $i < k$. This defines an extension $\mathcal{M}$ of $\mathcal{M}^f$.

Thanks to the Lemma 28, for every $\rho_s, \rho_r, \rho_\mathcal{O}$, for every $i \leq k$, $[\![t_P^i]\!]^{\sigma,\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}} = \mathcal{O}_P(\rho_s, m_0, \ldots, m_{i-1})$ and $[\![t_Q^i]\!]^{\sigma',\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}} = \mathcal{O}_Q(\rho_s, m'_1, \ldots, m'_{i-1})$. Now, according to our definition of $\widetilde{t_P}$ and thanks to the interpretation of g_i, for every $\rho_s, \rho_r, \rho_\mathcal{O}$, for every $i \leq k$, $[\![\widetilde{t_P^i}]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}} = \mathcal{O}_P(\rho_s, m_0, \ldots, m_{i-1})$ and $[\![\widetilde{t_Q^i}]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}} = \mathcal{O}_Q(\rho_s, m'_0, \ldots, m'_{i-1})$.

If we now consider the output of $\mathcal{A}^{\mathcal{O}}_{g_n}$, we have that, for every $\rho_r, \rho_\mathcal{O}$,

$$\mathcal{A}^{\mathcal{O}}_{g_n}([\![\widetilde{t_P^0}]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}}, \ldots, [\![\widetilde{t_P^k}]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}}, \eta, \rho_r) = \mathcal{B}^{\mathcal{O},\mathcal{O}_P}$$

and $\mathcal{A}^{\mathcal{O}}_{g_n}([\![\widetilde{t_Q^0}]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}}, \ldots, [\![\widetilde{t_Q^n}]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}}, \eta, \rho_r) = \mathcal{B}^{\mathcal{O},\mathcal{O}_Q}$. Thus, $\mathcal{M}^f$ and $\mathcal{A}^{\mathcal{O}}_{g_n}$ form a distinguisher on $\widetilde{t_P} \approx^{\mathcal{M}}_{\mathcal{O}} \widetilde{t_Q}$, which wins with the same probability as $\mathcal{B}$.

For the bottom to top direction, we are given a computational model $\mathcal{M}$ and a distinguisher $\mathcal{B}^{\mathcal{O}}$ on $\widetilde{t_P} \approx^{\mathcal{M}}_{\mathcal{O}} \widetilde{t_Q}$. We consider ϕ_i^P and σ_P as defined for $\widetilde{t_P}$, and ϕ_i^Q and σ_Q as defined for $\widetilde{t_Q}$.

Thanks to the Lemma 28, for every $\rho_s, \rho_r, \rho_\mathcal{O}$, for every $i \leq k$,

$$[\![t_P^i]\!]^{\sigma_P,\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}} = \mathcal{O}_P(\rho_s, [\![g_0()]\!]^{\sigma_P,\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}}, \ldots, [\![g_{i-1}(\phi_{i-1}^P)]\!]^{\sigma_P,\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}})$$

and $[\![t_Q^i]\!]^{\sigma_Q,\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}} = \mathcal{O}_Q(\rho_s, [\![g_0()]\!]^{\sigma_Q,\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}}, \ldots, [\![g_{i-1}(\phi_{i-1}^Q)]\!]^{\sigma_Q,\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}})$. Then, with the definition of $\widetilde{t_P}$, we have for every $\rho_s, \rho_r, \rho_\mathcal{O}$, for every $i \leq k$,

$$[\![\widetilde{t_P^i}]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}} = \mathcal{O}_P(\rho_s, [\![g_0()]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}}, \ldots, [\![g_{i-1}(\phi_{i-1}^P)]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}})$$

and $[\![\widetilde{t_Q^i}]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}} = \mathcal{O}_Q(\rho_s, [\![g_0()]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}}, \ldots, [\![g_{i-1}(\phi_{i-1}^Q)]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}})$.

We may now consider the PPTOM $B'^{\mathcal{O},\mathcal{O}_{P?Q}}$, which :

- Set m_0 to the result of $[\![g_0()]\!]^{\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}}$.
- For i going from 0 to $n-1$:
  - set t_i to the result of $\mathcal{O}_{P?Q}(m_i)$
  - set m_{i+1} to the result of $[\![g_{i+1}(t_0, \ldots, t_i)]\!]^{\sigma_P,\eta}_{\rho_s,\rho_r,\rho_\mathcal{O}})$
- set t_n to the result of $\mathcal{O}_{P?Q}(m_n)$
- outputs $\mathcal{B}^{\mathcal{O}}(t_0, \ldots, t_n)$

With our previous observation, $t_0, \ldots, t_n$ is either equal to $[\![\widetilde{t_P}]\!]^{\eta}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}$ or $[\![\widetilde{t_Q}]\!]^{\eta}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}$, and as $\mathcal{B}^{\mathcal{O}}$ is a distinguisher on $\widetilde{t_P} \approx^{\mathcal{M}}_{\mathcal{O}} \widetilde{t_Q}$, $B'^{\mathcal{O},\mathcal{O}_{P?Q}}$ is a distinguisher.

□

Lemma 22. *For any oracle $\mathcal{O}$ with support $\overline{n}$, the axiom $\forall k, k' \notin \overline{n}, k \sim k'$ is $\mathcal{O}$-sound.*

Proof. We are given a cryptographic library, and oracle $\mathcal{O}$ with support $\overline{n}$, and two names k, k' not in the support. We are given a $\mathcal{A}_{\mathcal{O}}$ which is a distinguisher over $k \sim k'$. We define a PPTTM $\mathcal{A}'$ which on input $(m, \rho_r, 1^{\eta})$:

- Splits ρ_r into three distinct infinite tapes ρ_{so}, ρ_{ra}, ρ_{ro}
- Simulates $\mathcal{A}^{\mathcal{O}(\rho_{so},\rho_{ro})}(m, \rho_{ra}, 1^{\eta})$

Let us a prove that $\mathcal{A}'$ is a distinguisher over $k \sim k'$, which contradicts the unconditional soundness of this axiom when there is no oracle.

We denote $\pi_{\overline{k}}(\rho_s, \eta)$ the tapes where every bit of ρ_s which does not correspond to a name of $\overline{k}$ is set to 0, and similarly $\pi_{\overline{k}^c}(\rho_s, \eta)$ where all bits for $\overline{k}$ are set to 0. We then have for any PPTOM $\mathcal{A}_{\mathcal{O}}$:

$$
\begin{aligned}
&|\mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_s,\rho_{\mathcal{O}})}([\![\overline{k}]\!]^{\sigma,\eta}_{\rho_s}, \rho_r, 1^{\eta}) = 1\} \\
&=^1 \mathbb{P}_{\rho_s,\rho_r,\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\pi_{\overline{k}}(\rho_s,\eta),\rho_{\mathcal{O}})}([\![\overline{n}]\!]^{\sigma,\eta}_{\pi_{\overline{k}^c}(\rho_s,\eta)}, \rho_r, 1^{\eta}) = 1\} \\
&=^2 \mathbb{P}_{\rho_{s1},\rho_{s2},\rho_r,\rho_{\mathcal{O}}}\{\mathcal{A}^{\mathcal{O}(\rho_{s1},\rho_{\mathcal{O}})}([\![\overline{n}]\!]^{\sigma,\eta}_{\rho_{s2}}, \rho_r, 1^{\eta}) = 1\} \\
&=^3 \mathbb{P}_{\rho_{so},\rho_s,\rho_{ra},\rho_{ro}}\{\mathcal{A}^{\mathcal{O}(\rho_{so},\rho_{ro})}([\![\overline{k}]\!]^{\sigma,\eta}_{\rho_s}, \rho_{ra}, 1^{\eta}) = 1\} \\
&=^4 \mathbb{P}_{\rho_s,\rho_r}\{\mathcal{A}'([\![\overline{k}]\!]^{\sigma,\eta}_{\rho_s}, \rho_r, 1^{\eta}) = 1\}
\end{aligned}
$$

1. Thanks to the definition of support, the oracle answers the same on $\pi_{\overline{k}}(\rho_s, \eta)$ and ρ_s;
2. we split ρ_s in two, to replace independent tapes $\pi_{\overline{k}}(\rho_s, \eta)$ and $\pi_{\overline{k}^c}(\rho_s, \eta)$;
3. we rename random tapes;
4. by construction of $\mathcal{A}'$.

This shows that A' has the same advantage as $A_{\mathcal{O}}$ against $k \sim k'$, which concludes the proof. □