Oracle Simulation: a Technique for Protocol Composition with Long Term Shared Secrets

Hubert Comon
LSV, CNRS & ENS Paris-Saclay & Inria & Université Paris-Saclay

Charlie Jacomme
LSV, CNRS & ENS Paris-Saclay & Inria & Université Paris-Saclay

Guillaume Scerri
Université Versailles Saint-Quentin & Inria

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.

CCS CONCEPTS

• **Security and privacy** → *Formal security models*; *Logic and verification.*

KEYWORDS

formal methods; composition framework; computational model; CCSA model; long term shared secrets

ACM Reference Format:
Hubert Comon, Charlie Jacomme, and Guillaume Scerri. 2020. Oracle Simulation: a Technique for Protocol Composition with Long Term Shared Secrets. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), November 9–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3372297.3417229

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 [3, 4, 14–16, 25] and other game-based composition methods [9–11, 28] 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 [6]. 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 [32], 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 [11].

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 [26], 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 [5, 18, 30], 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 [1]) 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 [32] 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., [3, 28]) 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 [9–12]. 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 [9, 11, 12, 23, 26]. 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 [11], 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, [23] extends [12] 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 consider in the long version [17]), 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 [15] and continued in [4, 14, 25] 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 [27] for the UC framework, ideas continued in [26] for Diffie-Hellman key exchanges, where they notably provide a proof of the ISO 9798-3 [1] 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 [12, Appendix A].

In [3], 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 [22] 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 [21] are unclear. Some limitations of PCL are detailed in [20].

The compositional security of SSH, in the sense of [12], has been studied in [31]. 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 [13], where the implementation is derived from a secure modelling in CryptoVerif [8].

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 [27], single session games [12], key-independent reductions [11] 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 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 [6], 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:

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

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 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 $[\![\cdot]\!]^{\eta,\sigma}_{\rho_s}$ 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, $[\![n]\!]^{\eta}_{\rho_s} = \mathcal{A}_n(1^\eta, \rho_s)$ (extracts a bit-string of length η from the random tape ρ_s) and $[\![\mathsf{sign}(x, k)]\!]^{\eta,\{x\mapsto m\}}_{\rho_s} = \mathcal{A}_{\mathsf{sign}}(m, \mathcal{A}_k(1^\eta, \rho_s))$.

2.2 Syntax of the protocols

The summary of the protocol syntax is given in fig. 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,

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

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

Figure 1: The protocol algebra

where all channels are taken out of a set 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 $\bot$ 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 $\bot; P$ reduces to $\bot$. 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. For simplicity, this latter construction is not presented in the current paper (see [17] for details).

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

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

Example 2.1. Given a randomized encryption function enc, we let $P(c, x_1, x_2)$ be the protocol $\mathtt{in}(c, x).\mathtt{out}(c, \mathsf{enc}(x, x_1, x_2))$. 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:

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

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

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.

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

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

Figure 2: Operational Semantics (excerpt)

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 fig. 2. The full semantics can be found in [17]. The transition relation $\underset{\mathcal{A}}{\longrightarrow}$ 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 provide in fig. 3 one of the possible reductions, for some attacker $\mathcal{A}$ that first sends a message over channel c_1 and then c_2.

We assume *action determinism* of the protocols [19]: given an input message on a given channel, at most one of the threads 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

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

$$\emptyset, (P(c_1, r_1, sk), \emptyset) \| (P(c_2, r_2, sk), \emptyset)$$

$$\underset{\mathcal{A}}{\longrightarrow} \emptyset, (\mathsf{out}(c_1, \mathsf{enc}(x, r_1, sk), \{x \mapsto m\}) \| (P(c_2 r_2, sk), \emptyset)$$

$m = \mathcal{A}(\emptyset, \rho_r)$ `is the first input message computed by the attacker`

$$\underset{\mathcal{A}}{\longrightarrow} \phi, (P(c_2, r_2, sk), \emptyset)$$

$\phi = [\![\mathsf{enc}(x, r_1, sk)]\!]^{\eta, \{x \mapsto m\}}_{\rho_s}$ `is the interpretation of the output received by the attacker`

$$\underset{\mathcal{A}}{\longrightarrow} \phi, (\mathsf{out}(c_2, \mathsf{enc}(x, r_1, sk), \{x \mapsto m_2\})$$

$m_2 = \mathcal{A}(\phi, \rho_r)$ `is the second input message computed by the attacker`

$$\underset{\mathcal{A}}{\longrightarrow} (\phi, [\![\mathsf{enc}(x, r_2, sk)]\!]^{\eta, \{x \mapsto m_2\}}_{\rho_s}), \mathbf{0}$$

Figure 3: Reduction example

protocol $P\{i \mapsto 1\} \| \dots \| P\{i \mapsto N\}$ 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 oracle. At this stage, we need stateless oracles in order to be compositional. Let us explain 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 was 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 $\mathsf{enc}(m, r, k)$ 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 ρ_O for the oracle coins.

Formally, oracles take as input tuples $(\overline{m}, r, s)$ where $\overline{m}$ is a finite sequence of bit-strings, 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 ρ_O, ρ_s respectively, in a deterministic way: the randomness extracted from ρ_O is uniquely determined by $\overline{m}, r, s$ and the extractions for different values do not overlap. See appendix A for more details.

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. We present the oracle $O^{\mathsf{enc,dec}}_k$ that provides both an encryption and a decryption oracle for the key k. On input $< m, r, s >$, the oracle:

- returns $\perp$ if $s \neq 1$ (s can be used to select a specific key, and the oracle only provides access to a single one);
- samples a bit-string r of length η from ρ_O, from a position determined by the input;
- if $m = (\text{“}dec\text{”}, n)$, returns $[\![\mathsf{dec}(x, y, k)]\!]^{\{x \mapsto n, y \mapsto r\}}_{\rho_s}\}$
- if $m = (\text{“}enc\text{”}, n)$, returns $[\![\mathsf{enc}(x, y, k)]\!]^{\{x \mapsto n, y \mapsto r\}}_{\rho_s}\}$

The support of the oracle is $\{k\}$, the only name used inside it.

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 $\mathcal{A}^{O(\rho_s, \rho_O)}(\omega, \rho_r)$, where ω is the input data of $\mathcal{A}$, ρ_r is its random tape and ρ_s, ρ_O are the random tapes accessible to the oracle. These definitions extend to multiple oracles $\langle O_1, \dots, O_n \rangle$, prefixing the query with an index in $\{1, \dots, n\}$.

Note that once the oracle's random tape is fixed, we ensure that all our oracles are deterministic. While not strictly necessary, this ensures that the various parts of the adversary do not need to explicitly share states, as they can always recompute the answer to oracle calls. This greatly simplifies proofs.

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 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 machines that interact with 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 $O_{\langle P_1, \dots, P_n \rangle}$ first checks which P_i is queried (there is at most one such i, by action determinism) and then replies as O_{P_i}.

Finally, we may consider oracles that combine protocols oracles and stateless oracles. $\mathcal{A}^{\langle O_1, \dots, O_m \rangle, \langle O_{P_1}, \dots, O_{P_n} \rangle}$ is also written $\mathcal{A}^{O_1, \dots, O_m, O_{P_1}, \dots, O_{P_n}}$.

Definition 2.4. Given a cryptographic library $\mathcal{M}^f$, protocols P, Q and a stateless oracle O , P is *O-indistinguishable from Q*, which we write $P \cong_O Q$, when, for every PPTOM $\mathcal{A}$,

$$|\mathbb{P}_{\rho_s, \rho_r, \rho_O}\{\mathcal{A}^{O(\rho_s, \rho_O), O_P(\rho_s)}(1^\eta, \rho_r) = 1\}$$
$$-\mathbb{P}_{\rho_s, \rho_r, \rho_O}\{\mathcal{A}^{O(\rho_s, \rho_O), O_Q(\rho_s)}(1^\eta, \rho_r) = 1\}|$$

is negligible in η.

We will later see several examples of O-indistinguishability. Remark that the oracle only increases the capabilities of the attacker, and thus for any P, Q and O, $P \cong_O Q$ implies $P \cong Q$.

3 SIMULATABILITY

As an example, consider a protocol P that uses a shared encryption key k, and may only accept encrypted messages whose plaintext satisfy a condition T_P (e.g. tagged messages). Intuitively, this protocol may be composed with any context that uses the shared key to encrypt plaintexts that cannot be confused with messages intended for P. We capture this by proving this protocol while giving the adversary access to an oracle $\mathcal{O}$, where $\mathcal{O}$ lets the adversary encrypt any message as long as it does not satisfy T_P. Then if P is "secure", P composed with any context that can be simulated using only $\mathcal{O}$ to access k is "secure". We define now this notion of simulatability.

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 " inside 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 $\mathcal{O}_P$ inside $\mathcal{D}^{\mathcal{O},\mathcal{O}_P}$.

Definition 3.1. Given an oracle $\mathcal{O}$, a cryptographic library $\mathcal{M}_f$, a protocol P, PPTOMs $\mathcal{D}^{\mathcal{O},\mathcal{O}_P}(\rho_{r_{\mathcal{D}}}, 1^\eta)$ and $\mathcal{A}^{\mathcal{O}}(\cdots, 1^\eta)$, we define $\mathcal{D}[\mathcal{A}^{\mathcal{O}}]^{\mathcal{O}}(\rho_r, 1^\eta)$ as the PPTOM 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 $\mathcal{A}^{\mathcal{O}(\rho_s,\rho_O)}(\rho_{r_1}, \theta, 1^\eta)$ 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 simulates $\mathcal{A}^{\mathcal{O}}$ 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 he 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 3.2. Given a sequence of names $\overline{n}$, an oracle $\mathcal{O}$ with support $\overline{n}$, a cryptographic library $\mathcal{M}_f$, a protocol P, then, $\nu\overline{n}.P$ is *$\mathcal{O}$-simulatable* if and only if there exists a PPTOM $\mathcal{A}^{\mathcal{O}}_P$ such that for every PPTOM $\mathcal{D}^{\mathcal{O},\mathcal{O}_P}$, for every η, every $\overline{v} \in (\{0,1\}^\eta)^{|\overline{n}|}$, $c \in \{0,1\}^\star$,

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

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 shared secrets as $\mathcal{O}$ does. This notion is not intended to capture any security property.

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. The formal definition corresponding to this intuition, and the proof that is is equivalent to our main definition are provided in appendix B.

Example 3.3. We fix first $\mathcal{M}_f$ (in an arbitrary way). We consider the following handshake protocol, in which n, r, k, r' are names:

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

We consider the oracle $\mathcal{O}^{\mathsf{enc},\mathsf{dec}}_k$ from example 2.3. We can easily prove that $\nu k.A$ is $\mathcal{O}^{\mathsf{enc},\mathsf{dec}}_k$-simulatable, as the attacker can sample an arbitrary n', use the oracle to compute $\mathsf{enc}(n', r_1, k)$ (which as the same distribution as $\mathsf{enc}(n', r, k)$ for any fixed value of k) with the request $<$ ("*enc*", n), $1, 1 >$, and $\mathsf{dec}(x, k)$ with the request $<$ ("*dec*", x), $1, 1 >$.

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

Thanks to the definition of appendix B, proving simulatability is in practice a syntactic verification. For instance, $\nu k.P$ is $\mathcal{O}^{\mathsf{enc},\mathsf{dec}}_k$ -simulatable (example 2.3) if k only appears in P in key position of an encryption or a decryption, and all encryptions use fresh randoms. With this definition, simulatability is stable under composition operators. This allows to reduce the simulation of large processes to the simulation of simpler processes.

Theorem 3.4. *Given an oracle $\mathcal{O}$, protocols P, Q, and $\overline{n} = \mathcal{N}(P) \cap \mathcal{N}(Q)$, if $\nu\overline{n}.P$ and $\nu\overline{n}.Q$ are $\mathcal{O}$-simulatable, then $\nu\overline{n}.P\|Q$ and $\nu\overline{n}.P;Q$ are $\mathcal{O}$-simulatable.*

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. 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 [2]. Note that as our tag checking function is an arbitrary boolean function: tagging can be implicit, as illustrated in our applications in section 5.

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 3.5. Given a name sk and a tagging function T, we define:

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

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 3.6. Any protocol P whose signatures are all of the form if $T(t)$ then $\mathsf{sign}(t, sk)$ for some term t (that does not use sk) is immediately $\nu sk.P$ $O^{\mathsf{sign}}_{T,sk}$-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 O that signs tagged messages, he immediately wins the EUF-CMA game, "forging" a signature by a simple call to O. We thus define a tagged unforgeability game (EUF-CMA$_{T,sk}$), derived from the EUF-CMA game [24], where the adversary wins the game only if he is able to produce the signature of a message that is not tagged.

Definition 3.7. A signature scheme (Sign, Vrfy) is EUF-CMA$_{T,sk}$ secure for oracle O and interpretation of keys $\mathcal{A}_{sk}$ if, for any PPTOM $\mathcal{A}$, the game described in fig. 4 returns **true** with probability (over ρ_r, ρ_s, ρ_O) negligible in η.

```
Game EUF-CMA$^{\circ,\mathcal{A}}_{T,sk}(\eta, \rho_r, \rho_s, \rho_O)$:
  List ← []
  (pk, sk) ← ($[\![pk]\!]_{\rho_s}$, $[\![sk]\!]_{\rho_s}$)
  (m, $\sigma$) ← $\mathcal{A}^{O(\rho_s,\rho_O),\mathsf{Sign}}(\mathsf{pk}, \eta, \rho_r)$
  Return $\neg T(\mathsf{m}) \wedge \mathsf{Vrfy}(\mathsf{pk}, \mathsf{m}, \sigma) \wedge \mathsf{m} \notin \mathsf{List}$

Oracle Sign(m):
  List ← (m : List)
  $\sigma$ ← Sign(sk, m)
  Return $\sigma$
```

Figure 4: Game for Tagged Unforgeability (EUF-CMA$_{T,sk}$)

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 EUF-CMA$_{T,sk}$ game rather than the classical EUF-CMA game. This reasoning is valid as EUF-CMA implies EUF-CMA$_{T,sk}$ even in the presence of the corresponding oracle.

Proposition 3.8. *If a signature scheme* (Sign, Vrfy) *is EUF-CMA secure for keys given by $\mathcal{A}_{sk}$, then* (Sign, Vrfy) *is EUF-CMA$_{T,sk}$ secure for the oracle $O^{sign}_{T,sk}$ and the interpretation of keys $\mathcal{A}_{sk}$.*

4 MAIN COMPOSITION THEOREMS

We distinguish between two complementary cases. First, 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, 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.

4.1 Composition without state passing

Essentially, if two protocols P, Q are indistinguishable, they are still indistinguishable when running inside 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 O an oracle. Let $\overline{n} = \mathcal{N}(P\|Q) \cap \mathcal{N}(R\|S)$. If $P \cong_O Q$ and $\nu\overline{n}.R\|S$ is O-simulatable, then theorem 4.3 yields $P\|R \cong_O Q\|R$, $R;P \cong_O R;Q$ and $(R;P)\|S \cong_O (R;Q)\|S$.

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 fig. 1 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. We say that a hole is terminal if it is not followed by any sequential composition.

Example 4.2. In the three cases of example 4.1, in order to apply the next theorem, we respectively use as contexts $C[\_1] := \_1 \| R$, $C[\_1] := R; \_1$ and $C[\_1] := (R; \_1) \| S$.

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 4.3. *Given a cryptographic library $\mathcal{M}_f$ and an oracle $\mathcal{O}$, let $P_1, \dots, P_n, Q_1, \dots, Q_n$ be protocols and $C[\_1, \dots, \_n]$ be a context such that all their channels are disjoint, 0 some constant, $\overline{n}$ a sequence of names and $c_1, \dots, c_n$ fresh channel names. If*

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

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

In addition, the advantage of the adversary in distinguishing $C[P_1, \dots, P_n]$ and $C[Q_1, \dots, Q_n]$ is bounded by the advantage of the adversary in distinguishing $P_1 \| \cdots \| P_n$ and $Q_1 \| \cdots \| Q_n$, with a polynomial overhead on the runtime of the simulator. Note that the bound we obtain for the reduction is polynomial in the running time of the context. The intuition behind the proof of the Theorem is as follows, where we denote $\overline{C} := C[\mathtt{out}(c_1, 0), \dots, \mathtt{out}(c_n, 0)]$. Intuitively, $\overline{C}$ abstracts out the components P_i, only revealing which P_i is running at any time. We start by showing 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 simulatability of $\overline{C}$, 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 the context thanks to proposition B.2, the hypothesis of the theorem implies $\overline{C} \| P_1 \| \dots \| P_n \cong_{\mathcal{O}} \overline{C} \| Q_1 \| \dots \| Q_n$. The second part is where our notion of simulatability comes into play, and where it is essential to deal carefully with the shared secrets. The proof of theorem 4.3 can be found in Appendix C, and other proofs in [17].

Given a protocol P and a context C, for theorem 4.3 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 the context. Notably, if we take as oracle the protocol oracle corresponding to the context itself, we can trivially apply theorem 4.3 but proving $P \cong_{\mathcal{O}} Q$ amounts to proving $C[P] \cong C[Q]$.

4.1.1 Application to tagged protocols. We consider two versions of SSH, calling them $SSH2$ and $SSH1$, 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 $SSH2$ running in the context of $SSH1$, we can use theorem 4.3. We denote I the idealized version of $SSH2$, the desired conclusion is then $SSH2 \| SSH1 \cong I \| SSH1$. We use $C[\_1] = \_1 \| SSH1$, it is then sufficient to find an oracle $\mathcal{O}$ such that:

(1) $\nu sk.SSH1$ is $\mathcal{O}$-simulatable (the simulatability of C directly follows),
(2) $SSH2 \cong_{\mathcal{O}} I$

Defining T_{SSH1} as the function checking the prefix, $SSH1$ is trivially $\mathcal{O}^{\mathrm{sign}}_{T_{SSH1},sk}$ simulatable (see definition 3.5) as $SSH1$ does enforce the tagging checks. We thus let $\mathcal{O}$ be $\mathcal{O}^{\mathrm{sign}}_{T_{SSH1},sk}$.

Assuming that sign verifies the classical EUF-CMA axiom, by proposition 3.8, it also verifies the tagged version EUF-CMA$_{T_{SSH1},sk}$. To conclude, it is then sufficient to prove that $SSH2 \cong_{\mathcal{O}} I$ with a reduction to EUF-CMA$_{T_{SSH1},sk}$.

4.1.2 Example of 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 4.4. Let us consider once again the protocol $P(x_1, x_2) := \mathtt{in}(c, x).\mathtt{out}(c, \mathsf{enc}(x, x_1, x_2))$ of example 2.1. We assume that we have a function kdf, which, given a random input, generates a suitable key for the encryption scheme. With *seed* a random name, let $C[\_1] :=$ let $sk = \mathsf{kdf}(seed)$ in $\_1$. $C[\|^i P(r_i, sk)]$ provides an access to an encryption oracle for the key generated in C:

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

Based on the fact that $C[\mathtt{out}(c, sk)] \cong_{\mathcal{O}} C[\mathtt{out}(c, n)]$ for an oracle $\mathcal{O}$ and a fresh name n such that $\nu sk.P$ is $\mathcal{O}$-simulatable, we can conclude that $C[\|^i P(r_i, sk)] \cong C[\|^i P(r_i, n)]$.

A classical example is a key exchange, used to establish a secure channel. The situation is dual with respect to theorem 4.3: contexts must be indistinguishable and the continuation simulatable.

Theorem 4.5. *Let C, C' be n-ary contexts such that each hole is terminal, and let $P_1(\overline{x}), \dots, P_n(\overline{x})$ be parameterized protocols, such that all channel sets are pairwise disjoint. Given a cryptographic library $\mathcal{M}_f$, an oracle $\mathcal{O}$, $\overline{n} \supseteq \mathcal{N}(C) \cap \mathcal{N}(P_1, \dots, P_n)$, $\overline{t_1}, \dots, \overline{t_n}, \overline{t'_1}, \dots, \overline{t'_n}$ sequences of terms, if*

(1) $C[\mathtt{out}(c_1, \overline{t_1}), \dots, \mathtt{out}(c_n, \overline{t_n})] \cong_{\mathcal{O}} C'[\mathtt{out}(c_1, \overline{t'_1}), \dots, \mathtt{out}(c_n, \overline{t'_n})]$
(2) $\nu\overline{n}.\mathtt{in}(c_1, \overline{x}).P_1(\overline{x}) \| \dots \| \mathtt{in}(c_n, \overline{x}).P_n(\overline{x})$ is $\mathcal{O}$-simulatable

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

In addition, the advantage of the adversary in distinguishing $C[P_1(\overline{t_1}), \dots, P_n(\overline{t_n})]$ and $C[P_1(\overline{t'_1}), \dots, P_n(\overline{t'_n})]$ is bounded by the advantage of the adversary in distinguishing $\widetilde{C}$ and $\widetilde{C'}$, with a polynomial overhead on the runtime of the simulator.

For the sake of simplicity, we often omit channels. When we do so, we only assume that they are all distinct. The following example shows how theorems 4.3 and 4.5 can be used to derive the security of one session of a key exchange composed with a protocol.

Example 4.6. 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 $KE[\_1, \_2] := I; \_1 \| R; \_2$ 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 5. We use both theorems 4.3 and 4.5. Assume, with k a fresh name, that:

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

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 4.3 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 4.5, 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:

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

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.

The following Theorem allows to prove the security of an arbitrary number of sessions of a protocol, even when the number of sessions depends on the security parameter.

Theorem 4.7. *Let O_r, 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 $Q_i(\overline{x}, \overline{y}, \overline{z})$ be parameterized protocols, such that the set of indexed names of P and 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 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 O_r simulatable.
(3) $\overline{s}$ is disjoint of the support of O.
(4) $P_0(\overline{p}, \overline{lsid}_0^P) \cong_{O_r, O} Q_0(\overline{p}, \overline{lsid}_0^Q, \overline{s})$

then, $\|^i P_i(\overline{p}, \overline{lsid}_i^P) \cong_O \|^i Q_i(\overline{p}, \overline{lsid}_i^Q, \overline{s})$

To prove the security of an unbounded number of sessions in parallel, we can with this Theorem only prove the security of a single session (Hypothesis 4), if this proof holds when the attacker has access to an oracle that allows to simulate the other sessions of the protocol (Hypothesis 1,2). The extra oracle O allows to apply in sequence our theorems, and we thus require that this oracle does not interfere with the replication (Hypothesis 3).

The extra argument $\overline{s}$ of Q is not necessary for the above theorem. It is however useful in our applications, where Q is an idealized version. 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 5), the main idea is to first use theorem 4.7 to reduce the multi-session security of a key exchange or a communication channel to a single session, and then use theorems 4.3 and 4.5 to combine the multiple key exchanges and the multiple channels.

5 APPLICATION TO KEY EXCHANGES

Although our framework is not tailored to key exchanges or any specific property, we choose here to focus on its application to key exchanges. 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). A formal corollary can be found in the long version [17].

5.1 Our model of key exchange

In order to obtain injective agreement, key exchanges usually use fresh randomnesses 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 4.6, KE is a key exchange with possible continuations. In addition, we consider multiple copies of KE, indexed by i, and local sid for each copy:

$$KE_i[\_1, \_2] := I(lsid_i^I, id^I); \_1 \| R(lsid_i^R, id^R); \_2$$

id^X is the identity of X and $lsid^X$ represents the randomness that is be used by X to derive its local session identifier.

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} the received identity. Symmetrically, R binds the variables x^R, x^R_{lsid} and x^R_{id}. For simplicity, we only consider here the case of two fixed honest identities. The general case is considered in [17].

If we denote $P_i^I(x^I) \| P_i^R(x^R)$ the continuation meant to use the secret key derived in the key exchange, $KE_i[P_i^I(x^I), P_i^R(x^R)]$ 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

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

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 is expressed as $\|^{i \leq N} KE_i[P_i^I(x^I), P_i^R(x^R)] \cong \|^{i \leq N} KE_i[Q_i^I(x^I), Q_i^R(x^R)]$, provided that the adversary advantage is polynomial in N.

5.2 Proofs of composed key exchange security

Following the same applications of theorems 4.3 and 4.5 as in example 4.6, we decompose the problem 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.

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 $\overline{s} = \{lsid_i^I, lsid_i^R\}_{i \in \mathbb{N}}$ the set of all copies of the local session identifiers.

5.2.1 Protocol simulatability. Assume that there is an oracle $\mathcal{O}_{P,Q}$ such that $\nu\overline{p}.in(x^I).P_i^I(x^I) \| in(x^R).P_i^R(x^R)$ is $\mathcal{O}_{P,Q}$ simulatable (and a similar result replacing P with Q), then, thanks to theorem 3.4, $\nu\overline{p}.\|^i(in(x^I).P_i^I(x^I) \| in(x^R).P_i^R(x^R))$ is $\mathcal{O}_{P,Q}$ simulatable (and similarly for Q). This meets the condition (2) of theorem 4.5.

5.2.2 Key exchange simulatability. For the simulation of the key exchange context, we need N 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 4.3 with a context C with $2N^2$ holes. C is the parallel composition of N contexts and, as above, we use theorem 3.4 to get the condition (1) of theorem 4.3. Let KE_i' be[3]

$$\begin{aligned} KE_i[\ & \underset{1 \leq j \leq N}{\text{if}}\ x_{lsid}^I = lsid_j^R \text{ then } \mathsf{out}(\langle i, j \rangle) \text{ else } \bot, \\ & \underset{1 \leq j \leq N}{\text{if}}\ x_{lsid}^R = lsid_j^I \text{ then } \mathsf{out}(\langle i, j \rangle) \text{ else } \bot] \end{aligned}$$

$\overline{C}$ is then $\|^{i \leq N} KE_i'$ and C can be inferred by replacing each $\mathsf{out}(\langle i, j \rangle)$ with a hole. Then, assuming that $\nu\overline{p}.KE_i'$ is $\mathcal{O}_{ke}$ simulatable, we get, thanks to theorem 3.4 the condition (1) of theorem 4.3.

5.2.3 Security of the protocol. Our goal is $\|^i P_i(k_i) \cong_{\mathcal{O}_{ke}} \|^i Q_i(k_i)$. Based on theorem 4.7, we only need an oracle $\mathcal{O}_r$ so that:

1. $\forall\ 1 \leq i, \nu\overline{p}, k_i.P_0(k_i)$ is $\mathcal{O}_r$ simulatable,
2. $\forall\ 1 \leq i, \nu\overline{p}, k_i.Q_0(k_i)$ is $\mathcal{O}_r$ simulatable,
3. $\overline{s}$ is disjoint of the support of $\mathcal{O}_{ke}$,
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 4.7. The intuition is similar to the notion of Single session game of [12], 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.

5.2.4 Security of the key exchange. The security of the key exchange is more bothersome, 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:

$$\begin{aligned} & \|^{i \leq N} KE_i[\mathsf{out}(x^I), \mathsf{out}(x^R)] \cong_{\mathcal{O}_{P,Q}} \\ & \quad \|^{i \leq N}\ KE_i[\ \underset{1 \leq j \leq N}{\text{if}}\ x_{lsid}^I = lsid_j^R \text{ then } \mathsf{out}(k_{i,j}) \text{ else } \bot, \\ & \qquad\qquad \underset{1 \leq j \leq N}{\text{if}}\ (x_{lsid}^R = lsid_j^I) \text{ then } \mathsf{out}(k_{j,i}) \text{ else } \bot] \end{aligned}$$

where the advantage of the attacker is polynomial in N.

Using a classical cryptographic hybrid argument (detailed in [17]), 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 eq. (1) using theorem 4.5. It is expressed, with $state_i^X = \langle x^X, lsid_i^X, x_{lsid}^X \rangle$, as

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

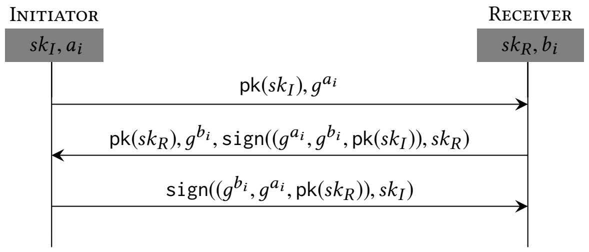
We further reduce the problem to proving the security of a single session even when there is an oracle simulating corrupted sessions. To this end, we need to reveal the dishonest local session's identifiers to the attacker, but also to allow him to perform the required cryptographic operations, e.g. 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 then only have to use theorem 4.3[4] with the following assumption: :

1. $\forall 1 \leq i \leq N-1, \nu lsid_i^I, id^I, lsid_i^R, id^R.$
   $KE_i[\mathsf{out}(x^I), \mathsf{out}(x^R)] \| \mathsf{out}(\langle lsid_i^R, lsid_i^I \rangle)$ is $\mathcal{O}_T$ simulatable.
2. $\overline{s}$ is disjoint of the support of $\mathcal{O}_{P,Q}$.
3. $KE_N[\mathsf{out}(\langle x^I, lsid_N^I, x_{lsid}^I \rangle), \mathsf{out}(\langle x^R, lsid_N^R, x_{lsid}^R \rangle)]$
   $\cong_{\mathcal{O}_T, \mathcal{O}_{P,Q}} KE_N[C_{I,R}, C_{R,I}]$

[3] we denote $\underset{1 \leq j \leq N}{\text{if}}\ c_i \text{ then } a_i \text{ else } a' := \text{if } c_1 \text{ then } a_1 \text{ else if } c_2 \cdots \text{ then } a_n \text{ else } a'$

[4] We also use theorem 3.4 to get the simulatability of N sessions in parallel from the simulatability of each session. In [17], this is formalized inside a dedicated Proposition.



INITIATOR
sk_I, a_i
RECEIVER
sk_R, b_i
$\mathsf{pk}(sk_I), g^{a_i}$
$\mathsf{pk}(sk_R), g^{b_i}, \mathsf{sign}((g^{a_i}, g^{b_i}, \mathsf{pk}(sk_I)), sk_R)$
$\mathsf{sign}((g^{b_i}, g^{a_i}, \mathsf{pk}(sk_R)), sk_I)$

Figure 5: ISO 9798-3 Diffie Hellman key exchange

$$\text{with } C_{X,Y} \ := \ \begin{cases} \text{if } x^X_{lsid} = lsid^Y_N \text{ then } \mathsf{out}(\langle k, lsid^X_N, x^X_{lsid}\rangle) \\ \text{else if } x^X_{lsid} \notin \bar{s}^Y \text{ then } \bot \\ \text{else } \mathsf{out}(\langle x^X, lsid^X_N, x^X_{lsid}\rangle) \end{cases}$$

Intuitively, if the initiator believes to be talking to the honest responder, then he 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 [12], 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 7.

6 BASIC DIFFIE-HELLMAN KEY EXCHANGE

We outline here the application of our framework to the ISO 9798-3 protocol, proven UC composable in [26]. We use our result to extend the security proof to a context with shared long term secrets. We present the protocol in fig. 5, and show how to instantiate the required values and oracles to perform the proof presented in section 5.2. The formal proofs (using the CCSA model [6]) are provided in [17].

The identity of each party is its long term signing 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 use g^{a_i} and g^{b_i} as $lsid^I_i$ and $lsid^R_i$. These values can also be used as implicit tagging since any signed message either depends on a_i or b_i. To define this implicit tagging, we extend the tagging function T of definition 3.5 so that it may depend on a second argument of arbitrary length, yielding $T(m, \bar{s})$, the corresponding signing oracle being denoted $O^{\mathsf{sign}}_{T,sk,\bar{s}}$. The exact definition is given later in definition 8.2. Letting $\bar{s}^I = \{a_i | i \geq 1\}$ and $\bar{s}^R = \{b_i | i \geq 1\}$, we define the implicit tagging functions T^I and T^R as $T^X(m, \bar{s}) := \exists s \in \bar{s}^X, \exists m_1, m_2.m = (m_1, g^s, m_2)$. With $O_T = O^{\mathsf{sign}}_{T^I,sk_I,\bar{s}}, O^{\mathsf{sign}}_{T^R,sk_R,\bar{s}}, O_{\bar{s}}$, where $O_{\bar{s}}$ simply reveals the elements in $\bar{s}$, we obtain the simulatability of multiple sessions of the key exchange (hypothesis 2 of section 5.2). For hypothesis 4 of section 5.2 we use the method of section 5.2.4: we consider the key exchange KE_0, followed either by the output of the computed keys, the computed $lsid$ and the real $lsid$ or by $C_{I,R}, C_{R,I}$.

The protocol sketched in fig. 5 actually assumes some verifications; for instance the value sent in the first message should match g^{a_i} in the last message. Therefore, when the protocol of fig. 5 is successfully completed, we can prove that if $x^I_{lsid} \neq g^{b_0}$, then $x^I_{lsid} \in \{g^{b_i} | i \in \mathbb{N}\}$, i.e. $T^I(x^R_{lsid})$ is true (and similarly for R). It follows that $C_{I,R}, C_{R,I}$ either corresponds to a matching conversation 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. The proof of the property 3) of section 5.2.4 is thus a real-or-random proof of a honestly produced key.

The previous security proof can be performed under oracle $O_{P,Q}$ that allows to simulate the continuation (hypothesis 1 of section 5.2). The continuation should be proven secure when using an ideal key (hypothesis 3 of section 5.2). 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. Without any shared secret, we do not need any oracle to simulate $in(k); L^I(k) \| in(k); L^R(k)$, we can choose a trivial $O_{P,Q}$ that does nothing.

7 EXTENSION TO KEY CONFIRMATIONS

We consider key exchanges where the key is derived in a first part of the protocol and then used in the second part. We proceed as in section 5, outlining how we may split the security proof into smaller proofs, using the same composition theorems at each step. Compared to [11], our method allows in addition long term secret sharing.

Consider a key exchange $I_i(lsid^I_i, id^I) \| R_i(lsid^R_i, id^R)$. We further split I and R into $I_i := I^0_i; I^1_i$ and $R_i := R^0_i; R^1_i$, where I^0_i and R^0_i correspond to the key exchange up to, but not including, the first use of the secret key, and I^1_i and R^1_i are the remaining parts of the protocol. The intuition behind the proof of security is that at the end of I^0_i and R^0_i, 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 and the key confirmation will fail. The key point for applying the composition theorems is that any protocol using the derived key must then be secure in the presence of an oracle O such that I^1_i, R^1_i is O simulatable (in particular the derived key is in the support of O). For typical cases, this is rather simply an oracle providing a hash of the key or some specific encryption of the confirmation message with the key. Specific hypotheses and details of the technique are provided in appendix D.

7.1 Application to SSH

SSH [32] is a protocol that allows a user to login onto a server from its platform. It is widely used in the version where signatures are used for authentication. An interesting feature is agent forwarding: once a user u is logged on a server S, he 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 the u's platform using the secure SSH channel between u and S. This represents a challenge: we compose a first key exchange with another one, the second one using a signature key already used in the first.

There is a known weakness in this protocol: any privileged user on S could use the agent 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, in Appendix, presents an example of a login followed by a login using the agent forwarding. 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 agent forwarding. 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.

We consider a scenario, in which there is an unbounded number of sessions of SSH, each with one (modified) agent forwarding, used to provide a secure channel for a protocol P. Thanks to multiple applications of theorems 4.3 and 4.5, 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 signature scheme and that the encryption is authenticated. P also has to satisfy the assumptions of appendix D.3. 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. Details can be found in [17]. 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.

8 FORMAL PROOFS IN THE CCSA MODEL

All the above proofs are cast in the CCSA model, which is well-suited for our purposes. In this model, formulas are first-order formulas built over a single predicate symbol $\sim$. Hypotheses on the cryptographic libraries are given as a (recursive) set of formulas A, called axioms. The protocols P, Q are folded into terms t_P, t_Q, in such a way that P is computationally indistinguishable from Q if $t_P \sim t_Q$ is derivable from A.

This result (computational soundness in [6]) also holds when the attacker has access to an oracle $\mathcal{O}$, provided that the formulas in A are sound with respect to such an attacker (in such a case, we say that the formulas are $\mathcal{O}$-sound). Details can be found in [17]. Therefore, we have to design axioms that are $\mathcal{O}$-sound. Let us give one such example, used in our proofs. Our definition of the tagging predicate may (here) depend on a possibly infinite list of secrets, as required in section 5 to verify that a session identifier belongs to the set of honest session identifiers.

Definition 8.1. Given a name sk, a sequence of names $\overline{s}$ and a function symbol T, we define the generic axiom scheme EUF-CMA$_{T,sk,\overline{s}}$ as, for any term t such that sk is only in key position:

$$\begin{array}{l}\mathtt{checksign}(t, pk(sk)) \;\Rightarrow \\ \quad T(\;\mathtt{getmess}(t), \overline{s}) \qquad \bigvee_{\mathtt{sign}(x,sk)\in\mathtt{St}(t)} \qquad (t = \mathtt{sign}(x, sk))\end{array}$$

The above formula is written in a simplified form (not using equivalences) and we do not define formally all its components. Its intuitive meaning is: if t is a valid signature with sk, then either the signed message satisfies the predicate T (the intention is "the message is correctly tagged") or it is a message that has been honestly produced in the past. The tagged signing oracle is defined as previously, only adding the extra argument to the tagging function.

Definition 8.2. Given a name sk, a sequence of names $\overline{s}$, and a predicate T, we define the generic signing oracle $\mathcal{O}^{sign}_{T,sk,\overline{s}}$ as follows:

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

Proposition 8.3. *For any computational model in which the interpretation of sign is EUF-CMA, any name sk, any sequence of names $\overline{s}$ such that $sk \notin \overline{s}$, and any polynomially computable interpretation of T, EUF-CMA$_{T,sk,\overline{s}}$ is $\mathcal{O}^{sign}_{T,sk,\overline{s}}$-sound.*

9 CONCLUSION

In summary, we designed a method that allows to decompose a security property of a compound protocol into security properties of its components. This works for parallel composition, but also sequential composition and replication: we designed a reduction from the security of multiples copies of a protocol to a security property of a single copy. Our method works even if the various components share secrets and (in case of sequential composition) when a state is passed to the other component. However, designing oracles can be a technical challenge of our model.

We illustrated the results with the applications to key exchanges and the SSH protocol with agent forwarding. Up to our knowledge these applications, in the general setting that we consider, cannot be covered by other methods. As future work, we plan to explore more complex properties of key exchanges and study other applications (MPC, e-voting), designing new applications of our theorems and new axioms when needed.

This approach is also well-suited for the formal proofs in the CCSA model. This model is designed for formal proofs that are computationally sound, but also works for stronger attackers, who typically can access some oracles. Though the $\mathcal{O}$-soundness proofs of the axioms, with standard cryptographic assumptions, have to be completed by hand, the core proofs can be completely formal and mechanizable. We plan to develop this mechanization.

In parallel, it could also be used to help proving complex protocols in EasyCrypt [7] for example, as security w.r.t. an attacker accessing an oracle can be formalized in this tool.

Acknowledgements. We wish to thank the anonymous reviewers for their useful comments, as well as Bogdan Warinschi and Ralf Küsters for interesting discussions. We are grateful for the support by the ANR under the TECAP grant (ANR-17-CE39-0004-01) and by the Institut Universitaire Français.

REFERENCES

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

[2] M. Arapinis, V. Cheval, and S. Delaune. 2012. Verifying Privacy-Type Properties in a Modular Way. In *2012 IEEE 25th Computer Security Foundations Symposium.* 95–109. https://doi.org/10.1109/CSF.2012.16

[3] Michael Backes, Markus Dürmuth, Dennis Hofheinz, and Ralf Küsters. 2008. Conditional reactive simulatability. *Int. J. Inf. Sec.* 7, 2 (2008), 155–169.

[4] Michael Backes, Birgit Pfitzmann, and Michael Waidner. 2007. The Reactive Simulatability (RSIM) Framework for Asynchronous Systems. *Inf. Comput.* 205, 12 (Dec. 2007), 1685–1720. https://doi.org/10.1016/j.ic.2007.05.002

[5] Gergei Bana, Rohit Chadha, and Ajay Kumar Eeralla. 2018. 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.* 350–372. https://doi.org/10.1007/978-3-319-98989-1_18

[6] Gergei Bana and Hubert Comon-Lundh. 2014. A Computationally Complete Symbolic Attacker for Equivalence Properties. In *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS'14)*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM Press, Scottsdale, Arizona, USA, 609–620. https://doi.org/10.1145/2660267.2660276

[7] Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella-Béguelin. 2011. Computer-Aided Security Proofs for the Working Cryptographer. In *Advances in Cryptology – CRYPTO 2011 (Lecture Notes in Computer Science)*, Vol. 6841. Springer, Heidelberg, 71–90.

[8] Bruno Blanchet. 2007. CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols. In *Dagstuhl seminar "Formal Protocol Verification Applied"*.

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

[10] Chris Brzuska, Antoine Delignat-Lavaud, Cédric Fournet, Konrad Kohbrok, and Markulf Kohlweiss. 2018. State Separation for Code-Based Game-Playing Proofs. In *ASIACRYPT (3) (Lecture Notes in Computer Science)*, Vol. 11274. Springer, 222–249.

[11] C. Brzuska, M. Fischlin, N. P. Smart, B. Warinschi, and S. C. Williams. 2013. Less is more: relaxed yet composable security notions for key exchange. *International Journal of Information Security* 12, 4 (Aug. 2013), 267–297. https://doi.org/10.1007/s10207-013-0192-y

[12] Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. 2011. Composability of Bellare-rogaway Key Exchange Protocols. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS '11)*. ACM, New York, NY, USA, 51–62. https://doi.org/10.1145/2046707.2046716

[13] David Cadé and Bruno Blanchet. 2013. From Computationally-Proved Protocol Specifications to Implementations and Application to SSH. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)* 4, 1 (March 2013), 4–31.

[14] Jan Camenisch, Stephan Krenn, Ralf Küsters, and Daniel Rausch. 2019. *iUC: Flexible Universal Composability Made Simple.* Technical Report.

[15] Ran Canetti. 2000. *Universally Composable Security: A New Paradigm for Cryptographic Protocols.* http://eprint.iacr.org/2000/067

[16] Ran Canetti and Tal Rabin. 2003. Universal Composition with Joint State. In *Advances in Cryptology - CRYPTO 2003 (Lecture Notes in Computer Science)*, Dan Boneh (Ed.). Springer Berlin Heidelberg, 265–281.

[17] Hubert Comon, Charlie Jacomme, and Guillaume Scerri. 2020. Oracle Simulation: a Technique for Protocol Composition with Long Term Shared Secrets - long version. https://hal.inria.fr/hal-02913866

[18] Hubert Comon and Adrien Koutsos. 2017. Formal Computational Unlinkability Proofs of RFID Protocols. In *Proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF'17)*, Boris Köpf and Steve Chong (Eds.). IEEE Computer Society Press, Santa Barbara, California, USA, 100–114. https://doi.org/10.1109/CSF.2017.9

[19] Véronique Cortier and Stéphanie Delaune. 2009. A method for proving observational equivalence. In *2009 22nd IEEE Computer Security Foundations Symposium*. IEEE, 266–276.

[20] Cas Cremers. 2008. On the Protocol Composition Logic PCL. In *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security (ASIACCS '08)*. ACM, New York, NY, USA, 66–76. https://doi.org/10.1145/1368310.1368324 event-place: Tokyo, Japan.

[21] Anupam Datta, Ante Derek, John C. Mitchell, Vitaly Shmatikov, and Mathieu Turuani. 2005. Probabilistic Polynomial-Time Semantics for a Protocol Security Logic. In *Automata, Languages and Programming (Lecture Notes in Computer Science)*, Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung (Eds.). Springer Berlin Heidelberg, 16–29.

[22] Nancy Durgin, John Mitchell, and Dusko Pavlovic. 2003. A Compositional Logic for Proving Security Properties of Protocols. *J. Comput. Secur.* 11, 4 (July 2003), 677–721. http://dl.acm.org/citation.cfm?id=959088.959095

[23] Marc Fischlin and Felix Günther. 2014. 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 (CCS '14)*. ACM, New York, NY, USA, 1193–1204. https://doi.org/10.1145/2660267.2660308 event-place: Scottsdale, Arizona, USA.

[24] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. 1988. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.* 17, 2 (1988), 281–308.

[25] Dennis Hofheinz and Victor Shoup. 2015. GNUC: A New Universal Composability Framework. *Journal of Cryptology* 28, 3 (July 2015), 423–508. https://doi.org/10.1007/s00145-013-9160-y

[26] Ralf Kusters and Daniel Rausch. 2017. A Framework for Universally Composable Diffie-Hellman Key Exchange. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Jose, CA, USA, 881–900. https://doi.org/10.1109/SP.2017.63

[27] Ralf Küsters and Max Tuengerthal. 2011. Composition Theorems Without Pre-established Session Identifiers. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS '11)*. ACM, New York, NY, USA, 41–50. https://doi.org/10.1145/2046707.2046715 event-place: Chicago, Illinois, USA.

[28] Ueli Maurer. 2011. Constructive Cryptography - A New Paradigm for Security Definitions and Proofs. In *TOSCA (Lecture Notes in Computer Science)*, Vol. 6993. Springer, 33–56.

[29] Kenneth G. Paterson, Jacob C. N. Schuldt, Martijn Stam, and Susan Thomson. 2011. On the Joint Security of Encryption and Signature, Revisited. In *Advances in Cryptology – ASIACRYPT 2011 (Lecture Notes in Computer Science)*, Dong Hoon Lee and Xiaoyun Wang (Eds.). Springer Berlin Heidelberg, 161–178.

[30] Guillaume Scerri and Stanley-Oakes Ryan. 2016. Analysis of Key Wrapping APIs: Generic Policies, Computational Security. IEEE Computer Society, 281–295. https://doi.org/10.1109/CSF.2016.27

[31] Stephen C. Williams. 2011. Analysis of the SSH Key Exchange Protocol. In *Cryptography and Coding (Lecture Notes in Computer Science)*, Liqun Chen (Ed.). Springer Berlin Heidelberg, 356–374.

[32] Tatu Ylonen and Chris Lonvick. [n. d.]. The Secure Shell (SSH) Transport Layer Protocol. https://tools.ietf.org/html/rfc4253

A STATELESS ORACLES

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

- a sequence of bit-strings $\overline{w} \in (\{0,1\}^*)^n$ and two bit-strings 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: $e_2(\overline{p}(s), \eta, \rho_s)$ is a sequence of bit-strings. It is also assumed to be injective.

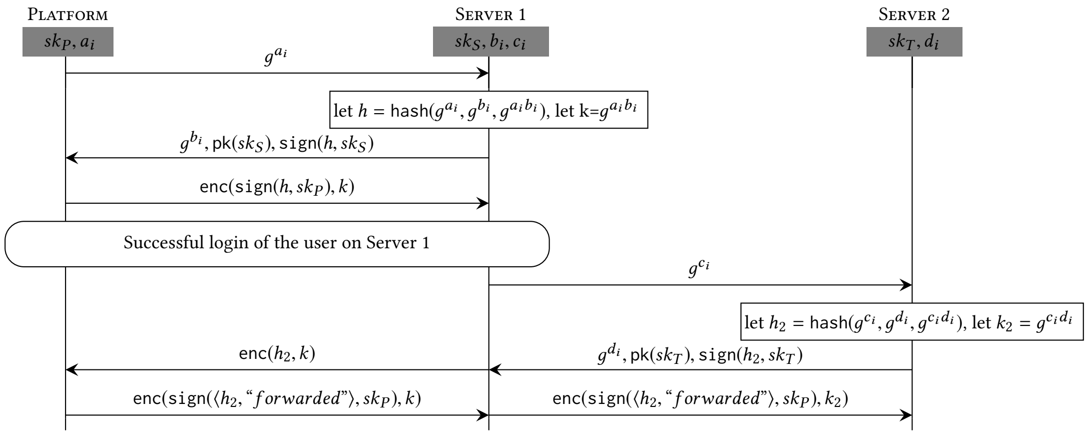
The third function takes $\eta, \overline{w}, r, s, e_1(n(\overline{w}, r), \eta, \rho_{\mathcal{O}}), e_2(\overline{p}(s), \eta, \rho_s)$ as input and returns a result (a bit-string) or a failure message.

B SIMULATABILITY

For technical reasons, we need to ensure that simulator's oracle calls and attacker's oracle calls do not use shared 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. This is illustrated in example B.3.

From now on, we replace the definition of simulatability by the following one:

Definition B.1. 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 PPTOM $\mathcal{A}^{\mathcal{O}}$ (using queries prefixed by 0) such that, for every $c \in \{0,1\}^\star$, for



Platform
sk_P, a_i
Server 1
sk_S, b_i, c_i
Server 2
sk_T, d_i
g^{a_i}
let h = hash($g^{a_i}, g^{b_i}, g^{a_i b_i}$), let k=$g^{a_i b_i}$
g^{b_i}, pk(sk_S), sign(h, sk_S)
enc(sign(h, sk_P), k)
Successful login of the user on Server 1
g^{c_i}
let h_2 = hash($g^{c_i}, g^{d_i}, g^{c_i d_i}$), let $k_2 = g^{c_i d_i}$
enc(h_2, k)
g^{d_i}, pk(sk_T), sign(h_2, sk_T)
enc(sign(⟨h_2, "forwarded"⟩, sk_P), k)
enc(sign(⟨h_2, "forwarded"⟩, sk_P), k_2)

Figure 6: SSH with agent forwarding

every $\overline{v} \in (\{0,1\}^\eta)^{|\overline{n}|}$, for every $m \geq 1$, for every PPTOM $\mathcal{B}^O$ (using random handles prefixed by 1),

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

where

$$\begin{aligned}\phi^2_{k+1} &= \phi^2_k, O_P(\rho_s,\theta^2_k)\\ \phi^1_{k+1} &= \phi^1_k, \mathcal{A}^{O(\rho_s,\rho_O)}(\rho_{r_1},\theta^1_k,\eta)\\ \theta^i_{k+1} &= \theta^i_k, \mathcal{B}^{O(\rho_s,\rho_O)}(\rho_{r_2},\eta,\phi^i_{k+1})\end{aligned}$$

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

The machine $\mathcal{A}$ 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.

This new definition of simulatability is equivalent to the one based on definition 3.1, as shown in proposition B.2. Alternative definitions for simulatability are discussed in appendix B.3.

Proposition B.2. *Given an oracle 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 PPTOM $\mathcal{A}^O_P$, $\nu\overline{n}.P$ is O-simulatable with $\mathcal{A}^O_P$ if and only if for every PPTOM $\mathcal{D}^{O,O_P,O_Q}$, for every η, every $\overline{v} \in (\{0,1\}^\eta)^{|\overline{n}|}$, $c \in \{0,1\}^\star$,*

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

The above characterization is the one we will use inside the proofs, where the extra protocol Q will be the context. The definition 3.1 was implicitly extended to support a distinguisher with an additional protocol oracle.

B.1 A detailed example of protocol simulation

The purpose of the following example is to illustrate the definitions and also to show the need for disjointness of the oracle tags of the attacker from the oracle tags of the simulator.

Example B.3. We take a more formal view on the example from example 3.3.

We let O be the encryption-decryption oracle: it expects an input ⟨"dec", m⟩ or ⟨"enc", m⟩, a key $s = 1$ (only one encryption key is considered), an input tag t and a security parameter η and returns

- enc(m,r,k) if the query is prefixed by "enc", k is the secret value extracted from ρ_s corresponding to the key 1, r is drawn from ρ_O and associated with the tag t (via e_1).
- dec(m,k) if the query is prefixed by "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 O-simulatable. (So, here, B is useless, and we let P be A).

O_P is then defined as follows (according to the section 2.5):

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

The machine $\mathcal{A}^O(\rho_{r_1},\theta,\eta)$ 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 O with (⟨"enc", α⟩ , 1, t) and gets back the bitstring $[\![\text{enc}(n,r,z)]\!]^{\eta,z\mapsto[\![k]\!]^\eta_{\rho_s}}_{\rho_{r_1},\rho_O}$. The interpretation of k is indeed fixed at once since it belongs to the "shared" names bounded by ν.
  3. outputs $[\![\text{enc}(x,r,z)]\!]^{\eta,x\mapsto\alpha,z\mapsto[\![k]\!]^\eta_{\rho_s}}_{\rho_{r_1}}$
- If $\theta = (m_1,m_2)$,
  1. calls O with (⟨"dec", m_2⟩ , 1, −) and gets back the bitstring $w = [\![\text{dec}(y,z)]\!]^{y\mapsto m_2,z\mapsto[\![k]\!]^\eta_{\rho_s}}$ 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 PPTOM $\mathcal{B}^O$.

- $\phi^1_1 = [\![\text{enc}(n,x,z)]\!]^{\eta,x\mapsto s_1,z\mapsto[\![k]\!]^\eta_{\rho_s}}_{\rho_{r_1}}$ where s_1 is the randomness used by 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 $\mathcal{O}$).

- $\phi_1^2 = [\![\mathsf{enc}(n, r, k)]\!]_{\rho_s}^{\eta}$
- $\theta_i^1 = w_i$, an arbitrary bit-string, computed by $\mathcal{B}^{\mathcal{O}}$ using the oracle $\mathcal{O}$, ϕ_i^1 and the random tape ρ_{r_2}.
- $\phi_2^1 = \phi_1^1, \mathsf{ok}$ if $[\![\mathsf{dec}(y, z)]\!]^{y \mapsto w_1, z \mapsto [\![k]\!]_{\rho_s}^{\eta}} = [\![\langle n, 1\rangle]\!]_{\rho_{r_1}}^{\eta}$ and an error otherwise
- $\phi_2^2 = \phi_1^2, \mathsf{ok}$ if $[\![\mathsf{dec}(x, k)]\!]_{\rho_s}^{\eta, x \mapsto w_2} = [\![\langle n, 1\rangle]\!]_{\rho_s}^{\eta}$ and an error otherwise

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

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

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 randomnesses 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 randomnesses used in the oracle queries of $\mathcal{B}$ are distinct from the randomnesses 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 $\mathsf{dec}(w_1, v), [\![\langle n, 1\rangle]\!]_{\rho_s}$ and $\mathsf{dec}(w_2, v), [\![\langle n, 1\rangle]\!]_{\rho_s}$, 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

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

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.

B.2 Prefixed model

As demonstrated in ther 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 B.4. Given a PPTOM $\mathcal{A}^{\mathcal{O}}$ 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 bit-strings.

The following lemma shows that we can, w.l.o.g., consider models, in which the tags are prefixed. It uses the formal definition of a stateless oracle of Appendix A.

Lemma B.5. *For any non-empty constant c and any PPTOM $\mathcal{A}^{\mathcal{O}}$, 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, without loss of generality, 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.

B.3 Alternative notions 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 do not provide the proof, but the intuition is that there exists a "universal" distinguisher, namely the PPTOM $\mathcal{D}$, 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 mimick the probabilistic behavior of the protocol.

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

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

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).

C PROOF OF THEOREM 4.3

We first generalize theorem 4.3, so that it is more easily usable. Notably, rather than restricting the channels of the protocols and the context to be disjoint, we allow to define a renaming between channels.

Theorem C.1. *Let $C[\_1, \dots, \_n]$ be a context. Let $P_1, \dots, P_n$, $Q_1, \dots, Q_n$ be protocols, and let $\sigma : C(P_1, \dots, P_n) \mapsto C$ 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 O, with $\overline{n} \supseteq \mathcal{N}(C) \cap \mathcal{N}(P_1, \dots, P_n, Q_1, \dots, Q_n)$, if $\nu\overline{n}.\overline{C}$ is O-simulatable and $P_1\| \dots \|P_n \cong_O Q_1\| \dots \|Q_n$, then*

$$C[P_1\sigma, \dots, P_n\sigma] \cong_O C[Q_1\sigma, \dots, Q_n\sigma]$$

C.1 Oracle simulation

We first show that O-simulation, whose definition implies the identical distributions of two messages produced either by te simulator of by the oracle, implies the equality of distributions of message sequences produced by either the oracle or the simulator.

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 B.1.

Lemma C.2. *Given a cryptographic library $\mathcal{M}_f$, a sequence of names $\overline{n}$, an oracle O with support $\overline{n}$ and a protocol P, that is O-*simulatable *with $\mathcal{A}^O$, 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 PPTOM $\mathcal{B}^O$ (using tags prefixed by 1):*

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

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

The proof of this lemma can be found in [17].

We now prove that definition B.1 implies definition 3.1, i.e that the simulatability implies that we can replace a protocol oracle by its simulator.

Lemma C.3. *Given an oracle O (with support $\overline{n}$), a cryptographic library $\mathcal{M}_f$, a sequence of names $\overline{n}$, P, Q protocols, s.t $\nu\overline{n}.P$ is O-simulatable with $\mathcal{A}^O_P$ and $\mathcal{N}(P) \cap \mathcal{N}(Q) \subseteq \overline{n}$ then, for every PPTOM D^{O,O_P,O_Q} (prefixed by 1), every η, every $\overline{v} \in D^\eta_{\overline{n}}$ and every $c \in \{0,1\}^\star$,*

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

The idea is to use the definition of O-simulatablity, using a PPTOM $\mathcal{B}^O$ 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 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 O-simulation). The other randomness in Q can be drawn by $\mathcal{B}$ from ρ_r, without changing the distribution of O_Q's replies.

Note that the Lemma also implies that:

$$\begin{aligned} &\mathbb{P}_{\rho_s, \rho_r, \rho_O}\{\mathcal{D}^{O,O_P,O_Q}(\rho_r, 1^\eta) = c\} \\ &= \mathbb{P}_{\rho_s, \rho_r, \rho_O}\{\mathcal{D}[\mathcal{A}^O_P]^{O,O_Q}(\rho_r, 1^\eta) = c\} \end{aligned}$$

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 O_P (or when $\mathcal{D}$ halts if this occurs before the mth call) and returning the last query to O_P.

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

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

Part 1

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

$$\begin{aligned} &\mathbb{P}_{\rho_s, \rho_r, \rho_O}\{\mathcal{D}^{O,O_P}(\rho_r, 1^\eta) = c\} \\ &= \mathbb{P}_{\rho_s, \rho_r, \rho_O}\{\mathcal{D}[\mathcal{A}^O_P]^O(\rho_r, 1^\eta) = c\} \end{aligned}$$

This is a straightforward consequence of lemma C.2. Writing respectively $p^1_1(c) = \mathbb{P}_{\rho_s, \rho_r, \rho_O}\{\mathcal{D}^{O,O_P}(\rho_r, 1^\eta) = c\}$ and $p^2_1(c) = \mathbb{P}_{\rho_s, \rho_r, \rho_O}\{\mathcal{D}[\mathcal{A}^O_P]^O(\rho_r, 1^\eta) = c\}$, Using ρ_{r_1}, ρ_{r_2} as in definition 3.1, we have

$$\begin{aligned} p^1_1(c) \quad = \quad &\sum_{r_{\mathcal{B}}, r_2} \mathbb{P}_{\rho_s, \rho_r, \rho_O}\{\mathcal{D}^{O,O_P}(\rho_r, 1^\eta) = c \\ &\qquad | ([\![\overline{n}]\!]^\eta_{\rho_s}, \rho^{\mathcal{B}}_O, \rho_{r_2}) = (\overline{v}, r_{\mathcal{B}}, r_2)\} \\ &\qquad \times \mathbb{P}_{\rho_s, \rho_r, \rho_O}\{([\![\overline{n}]\!]^\eta_{\rho_s}, \rho^{\mathcal{B}}_O, \rho_{r_2}) = (\overline{v}, r_{\mathcal{B}}, r_2)\} \\ p^2_1(c) \quad = \quad &\sum_{r_{\mathcal{B}}, r_2} \mathbb{P}_{\rho_s, \rho_r, \rho_O}\{\mathcal{D}[\mathcal{A}^O_P]^O(\rho_r, 1^\eta) = c \\ &\qquad | ([\![\overline{n}]\!]^\eta_{\rho_s}, \rho^{\mathcal{B}}_O, \rho_{r_2}) = (\overline{v}, r_{\mathcal{B}}, r_2)\} \\ &\qquad \times \mathbb{P}_{\rho_s, \rho_r, \rho_O}\{[\![\overline{n}]\!]^\eta_{\rho_s} = \overline{v}, \rho^{\mathcal{B}}_O = r_{\mathcal{B}}, \rho_{r_2} = r_2\} \end{aligned}$$

We let

$$\begin{aligned} p^1_2(r_{\mathcal{B}}, r_2, \bar{v}, c) = \mathbb{P}_{\rho_s, \rho_r, \rho_O}\{&\mathcal{D}^{O,O_P}(\rho_r, 1^\eta) = c \\ &| ([\![\overline{n}]\!]^\eta_{\rho_s}, \rho^{\mathcal{B}}_O, \rho_{r_2}) = (\overline{v}, r_{\mathcal{B}}, r_2)\} \end{aligned}$$

and

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

We use definition B.1 with $\mathcal{B}^O(\rho_{r_2}, \eta, \phi)$ as the machine that simulates $\mathcal{D}_m$ for $m = |\phi|$ and using ϕ instead of querying the oracle. Let us define ϕ_m^i, θ_m^i for $i = 1, 2$ as in definition B.1. 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_m^i be the last message of θ_m^i. By definition of $\mathcal{D}$ and $\mathcal{B}$ we have $v_m^1 = v_m$ and $v_m^2 = v'_m$. Choosing m such that $\mathcal{D}$ makes less than m oracle calls, we have

$$p_2^i(r_{\mathcal{B}}, r_2, \bar{v}, c) = \\ \sum_{\bar{x} \text{ s.t. } x_m = c, \bar{y}} \mathbb{P}_{\rho_s,\rho_{r_1},\rho_{r_2},\rho_O}\{\theta_m^i = \overline{x}, \phi_m^i = \overline{y} \\ \mid ([\![\overline{n}]\!]_{\rho_s}^{\eta}, \rho_O^{\mathcal{B}}, \rho_{r_2}) = (\overline{v}, r_{\mathcal{B}}, r_2)\}.$$

lemma C.2 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:

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

We are thus going to show that, with the interpretation of $\overline{n}$ fixed, we can simulate O_Q inside some $\mathcal{D}'$ by sampling inside ρ_r instead of ρ_s. However, both computations of O_P and 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 $\mathcal{M}^{O,O_Q}$, we let $[\mathcal{M}]_{\overline{n}}^{O}$ be the machine that executes $\mathcal{M}$, simulating 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 $O_Q(\rho_s, \theta) := O_Q((\rho_{s_0}, \rho_{s_1}, \rho_{s_2}), \theta)$ 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 $[\mathcal{M}]_{\overline{n}}^{O}(\rho_r, 1^\eta)$ 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 O_Q with input u, the machine adds u to θ, and produces the output of $O_Q((\overline{v}, \rho_{rQ}, 0), \theta)$.

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

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

Since

(1) O_Q does not access ρ_{s_2}
(2) O_P does not access ρ_{s_1}
(3) We are sampling under the assumption that $[\![\overline{n}]\!]_{\rho_s}^{\eta} = \overline{v}$, 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:

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

By applying the left-handside of (1) to $[\mathcal{D}]_{\overline{n}}^{O,O_P(\rho_s)}(\rho_r, 1^\eta)$ and $[\mathcal{D}[\mathcal{A}_P^O]_j]_{\overline{v}}^{O}(\rho_r, 1^\eta)$, 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.

□

C.2 Proof of the Theorem

Recall that we denote $\overline{C}$ the context 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.

For any protocols P, Q, we denote $\mathrm{Adv}_{\mathcal{A}}^{P \cong Q}(t)$ the advantage (c.f. definition 2.4) for the PPTOM $\mathcal{A}$ (with potential access to an oracle) with execution time bounded by t

Theorem C.1. *Let $C[\_1, \ldots, \_n]$ be a context. Let $P_1, \ldots, P_n$, $Q_1, \ldots, Q_n$ be protocols, and let $\sigma : C(P_1, \ldots, P_n) \mapsto C$ such that $\overline{C} \| P_1 \| \ldots \| P_n$, $\overline{C} \| Q_1 \| \ldots \| Q_n$, $C[P_1\sigma, \ldots, P_n\sigma]$, $C[Q_1\sigma, \ldots, Q_n\sigma]$ are protocols. Given a cryptographic library $\mathcal{M}^f$, an oracle O, with $\overline{n} \supseteq \mathcal{N}(C) \cap \mathcal{N}(P_1, \ldots, P_n, Q_1, \ldots, Q_n)$, if $\nu\overline{n}.\overline{C}$ is O-simulatable and $P_1 \| \ldots \| P_n \cong_O Q_1 \| \ldots \| Q_n$, then*

$$C[P_1\sigma, \ldots, P_n\sigma] \cong_O C[Q_1\sigma, \ldots, Q_n\sigma]$$

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

$$C[P_1\sigma, \ldots, P_n\sigma] \cong_O C[Q_1\sigma, \ldots, Q_n\sigma].$$

We first build an attacker against

$$\overline{C} \| P_1 \| \ldots \| P_n \cong_O \overline{C} \| Q_1 \| \ldots \| Q_n.$$

Let us construct $\mathcal{B}^{O,O_{\overline{C}},O_{R_1},\ldots,O_{R_n}}$ with either for every i, $R_i = P_i$, or, for every i, $R_i = Q_i$. $\mathcal{B}^{O,O_{\overline{C}},O_{R_1},\ldots,O_{R_n}}$ 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 $\mathcal{A}^{O,O_{C[R_1\sigma,\ldots,R_n\sigma]}}$ but, each interaction with $O_{C[R_1\sigma,\ldots,R_n\sigma]}$ and the corresponding request (c, m) is replaced with:

- if there exist i such that $c_i = 1$ and $c \in C(R_i\sigma)$ then
  - query O_{R_i} with $(c\sigma^{-1}, m)$

- if O_{R_i} returns $\perp$, then, if contexts C_1 and C_2 are such that $C[\_1, \dots, \_n] = C_1[\_i; C_2]$, it adds to $\overline{x}$ the channels $C(C_2)$. (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 $c \in C(\overline{C})$ and $c \notin \overline{x}$ then
  - query $O_{\overline{C}}$ with (c, m)
  - if $O_{\overline{C}}$ answers $\top$ on channel γ_i, set $c_i = 1$
  - else continue with the reply of $O_{\overline{C}}$

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 exist i such that $c_i = 1$ and $c \in C(R_i)$ is always satisfied by a unique i, otherwise $C[P_1\sigma, \dots, P_n\sigma]$ or $C[Q_1\sigma, \dots, Q_n\sigma]$ 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 such that the runtime of $\mathcal{B}$ is bounded (uniformly in $C, P_1, \dots, P_n, Q_1, \dots Q_n$) by $p_S(n, t, |C|, |\sigma|)$:

$$\begin{aligned}\mathrm{Adv}_{\mathcal{A}^O}^{C[P_1,\dots,P_n] \cong C[Q_1,\dots,Q_n]}(t) \\ \leq \mathrm{Adv}_{\mathcal{B}^{O,O_{\overline{C}}}}^{P_1 \| \dots \| P_n \cong Q_1 \| \dots \| Q_n}(p_S(n, t, |C|, |\sigma|))\end{aligned}$$

Now, with the fact that $\nu\overline{n}.\overline{C}$ is O simulatable, we have a simulator $\mathcal{A}_{\overline{C}}^{O}$ such that, thanks to lemma C.3, $\mathcal{B}[\mathcal{A}_{\overline{C}}^{O}]^{O,O_R}$ behaves exactly as $\mathcal{B}^{O,O_{\overline{C}},O_R}$.

We have, for p_C the polynomial bound on the runtime of $\mathcal{A}_{\overline{C}}$, by definition 3.1,

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

and finally,

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

□

D PROOFS WITH KEY CONFIRMATIONS

D.1 Key exchange and protocol simulatability

We modify slightly the hypotheses of sections 5.2.1 and 5.2.2 to reflect the fact that we now consider the key confirmation to be part of the continuation:

(1) $\nu\overline{p}.in(x).I^1(x); P^I(x)$, $in(x).R^1(x); P^R(x)$, $in(x).I^1(x); Q^I(x)$, $in(x).R^1(x); Q^R(x)$ are $O_{P,Q}$ simulatable.

(2) $\nu\overline{p}.\ \|^{i \leq N}\ I_i^0(lsid_i^I, id^I);\ \underset{1 \leq i \leq N}{\text{if}}\ x_{lsid}^I = lsid_j^R$ then
$\quad$ out$(\langle i, j\rangle)$
$\quad$ else $I_i^1(x^I); \perp$
$\|^{i \leq N}\ R_i^0(lsid_i^R, id^R);\ \underset{1 \leq i \leq N}{\text{if}}\ x_{lsid}^R = lsid_j^I$ then
$\quad$ out$(\langle i, j\rangle)$
$\quad$ else $R_i^1(x^R); \perp$

is O_{ke} simulatable.

D.2 Security of the protocol

Compared to section 5.2.3, the continuation must be secure even in the presence of the messages produced during the key confirmation:

$$\begin{aligned}\|^{i \leq N} I_i^1(x^I); P_i^I(x^I) \| R_i^1(x^R); P_i^R(x^R) \cong_{O_r, O_k} \\ \|^{i \leq N} I_i^1(x^I); Q_i^I(x^I) \| R_i^1(x^R); Q_i^R(x^R)\end{aligned}$$

We can once again split this goal into a single session proof using theorem 4.7. 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.

D.3 Security of the key exchange

We proceed in a similar way as in section 5.2.4 and we use the same notations. Let us define

$$KE_i^0[\_1, \_2] := I_i^0(lsid_i^I, id^I); \_1 | R_i^0(lsid_j^R, id^R); \_2$$

The following Hypothesis are then suitable:

(1) $\forall i \leq N, \nu lsid_i^I, id^I, lsid_i^R, id^R.$
$KE_i^0[\mathsf{out}(x^I), \mathsf{out}(x^R)] \| \mathsf{out}(\langle lsid_i^R, lsid_i^I\rangle)$ is O_T-simulatable
(2) $\overline{s}$ is disjoint of the support of $O_{P,Q}$.
(3) $KE^0[D_I, D_R] \cong_{O_{KE}, O_{P,Q}} KE^0[C_{I,R}, C_{R,I}]$

$$\begin{aligned}\text{where}\quad D_X :=\ & \text{if } x_{lsid}^X \notin \overline{s}^Y \text{ then } X^1(x^X) \\ & \text{else } \mathsf{out}(\langle x^X, lsid_0^X, x_{lsid}^X\rangle) \\ C_{X,Y} :=\ & \text{if } x_{lsid}^X = lsid_0^Y \text{ then } \mathsf{out}(\langle k, lsid_0^X, x_{lsid}^X\rangle) \\ & \text{else if } x_{lsid}^X \notin \overline{s}^Y \text{ then } X^1(x^X); \mathsf{out}(\perp) \\ & \text{else } \mathsf{out}(\langle x^X, lsid_0^X, x_{lsid}^X\rangle)\end{aligned}.$$

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, a party must always be partnered with some honest session, i.e $x_{lsid}^X \notin \overline{s}^Y$ never occurs. When two honest parties are partnered, but are not the singled out parties, they leak their states.