Composing protocols with randomized actions

Matthew S. Bauer[1], Rohit Chadha[2], and Mahesh Viswanathan[1]

[1] University of Illinois at Urbana-Champaign
[2] University of Missouri

Abstract. Recently, several composition results have been established, showing that two cryptographic protocols proven secure against a Dolev-Yao adversary continue to afford the same security guarantees when composed together, provided the protocol messages are tagged with the information of which protocol they belong to. The key technical tool used to establish this guarantee is a separation result which shows that any attack on the composition can be mapped to an attack on one of the composed protocols running in isolation. We consider the composition of protocols which, in addition to using cryptographic primitives, also employ randomization within the protocol to achieve their goals. We show that if the protocols never reveal a secret with a probability greater than a given threshold, then neither does their composition, given that protocol messages are tagged with the information of which protocol they belong to.

1 Introduction

The design of correct cryptographic protocols is a highly non-trivial task, and security flaws are often subtle. Attacks on many protocols that were previously "proved" secure by hand, have been discovered. One approach that improves the confidence in the correctness of security protocols is formal analysis. In order to make the analysis amenable to automation, usually the assumption of perfect cryptography is made. In this "Dolev-Yao" framework, protocol messages are symbolic terms identified modulo an equational theory (and not bit-strings) that model cryptographic operations. Security is then proven in the presence of an omnipotent attacker that can read all messages sent on public channels by protocol participants, remember the (potentially unbounded) communication history, and (nondeterministically) inject new messages in the network addressed to particular participants while remaining anonymous. This Dolev-Yao model has shown to be very successful in identifying security flaws.

Cryptographic protocols are often proven secure in isolation. In practice, however, they may be executing concurrently or sequentially, in a modular fashion, with other protocols. For example, a number of security protocols involve a sub-protocol in which short-term secret keys are exchanged. While analyzing such protocols, often the sub-protocol is abstracted away by assuming that the protocol participants have successfully shared secrets. However, two cryptographic protocols proven secure independently may not remain secure if they are executed compositionally. The central problem is that these protocols may share some secret data, as in the key exchange situation described above.

Hence, a number of recent papers have identified sufficient conditions under which such protocol compositions can be proven secure — safety properties are considered in [19, 29, 27, 28, 2, 16, 17, 5, 21, 15, 18, 30, 3] and indistinguishability properties in [4, 3], while [19, 29, 27] provide a general framework for proving that protocols compose securely. Other papers [2, 28] essentially show that protocol compositions are secure if messages from one protocol cannot be confused with messages from another protocol. [10] shows that this continues to be the case even when dishonest participants do not tag their messages properly. This can be ensured if certain protocol transformations are made (see for example [16, 17, 21, 4]). Essentially, these protocol transformations require that all protocol messages are *tagged* with the protocol name and protocol instance to which they belong. The exact choice of tagging scheme depends on the desired security property; incorrect tagging can actually make a secure protocol insecure [21]. In the computational model, the problem of composing protocols securely has been studied in [8, 9].

The focus of this paper is to extend this work on secure protocol composition to protocols that employ randomization. Randomization plays a key role in the design of algorithmic solutions to problems arising in distributed computing and security. For example, randomization is essential in implementing cryptographic primitives such as encryption and key generation. Randomization is also used in cryptographic protocols to achieve security guarantees such as fair exchange (see [31, 6, 22]), anonymity (see [13, 32, 24]), voter privacy in electronic voting (see [33]) and denial of service prevention (see [26]).

We study the problem of when the composition of a (randomized) sub-protocol P followed by (randomized) sub-protocol Q is secure. For non-randomized protocols, this problem was studied in [18]. Our composition framework generalizes that of [18] to handle sequential, parallel and a form of vertical composition while extending to randomized protocols. They show that if one can prove that P and Q do not reveal shared secrets when run in isolation (in that case Q is assumed to generate fresh secret keys), then the sequential composition of P and Q does not reveal any secret of Q if the protocol messages are tagged with the information of which protocol they belong to. The key technical tool used to establish this guarantee is a separation result which shows that any attack on the composition can be mapped to an attack on one of P or Q. This is achieved by first showing that, as the protocol messages are tagged, messages from one protocol cannot be confused with the messages of the other protocol. Then an attack trace can be simply separated into traces of P and Q. We study the same problem for the case when P and Q are randomized protocols. The protocols themselves are expressed in a variant of the probabilistic applied-pi calculus [25] which is an extension of applied pi-calculus [1]. The Probabilistic applied-pi calculus is a convenient formalism to describe and analyze randomized security protocols in the Dolev-Yao model.

Contributions: Our first composition result is for the composition of one session of P and one session of Q. We show that if P (in isolation) is secure with probability at least p (i.e., the shared secrets are not leaked) and Q is secure with probability at least q, then the composed protocol is secure with probability at least pq, provided the protocol messages are tagged with the information of protocol to which they belong. Although we exploit some techniques used in [18] to establish this result, there are important differences. This is due to the fact that their separation result does not carry over to the randomized setting. Essentially, this is because an attack on the composition of P and Q is no longer a trace, but is instead a tree, as the protocol itself makes random choices. As a consequence, in different branches representing different resolutions of the randomized coin tosses, it is possible that the attacker may choose to send different messages (See Example 4 on Page XXIV). In such a case, an attack on the composition of P and Q cannot be separated into an attack on P and an attack on Q.

Another challenge manifested in the context of randomized protocols is that one must consider adversaries whose actions do not depend on the result of private coin tosses made by protocol participants, as observed in [20, 14, 7, 23, 12, 11]. For example, consider the process that outputs two nonces n_1 and n_2 and then flips a fair coin. With $\frac{1}{2}$ probability, the process takes an input and tests if the input was n_1 (resp. n_2). If either test passes, the process outputs a secret. When this process in analyzed with respect to the class of all schedulers, the secret can be derived with probability 1. The adversary simply chooses the input based on the result of the protocol's coin toss. For the majority of cryptographic protocols employing randomness, such attacks are not considered, as their security is predicated on the privacy of the protocols coin tosses. To accommodate this, we restrict the class of attackers by developing a notion of an attackers *view* of a protocol execution, and mandate that whenever an attacker has an identical view for two different branches (executions) of a protocol, its input must be the same. This notion is adopted from [20, 14, 7, 23, 12, 11], and is the first formalization of this concept within the applied-pi calculus. Considering a more restricted class of adversaries imposes additional challenges in our setting, as membership is this sub-class of adversaries must be maintained when mapping attack traces on composed protocols to attack traces on the individual protocols constituting the composition. As demonstrated in Example 1 on Page XI, this class of adversaries allows privacy guarantees, typically modeled as indistinguishability properties, to instead be modeled as reachablility properties.

Our second composition result concerns multiple sessions of the composed protocol. Here, we would like to show that if n sessions of P are secure with probability at least p and n sessions of Q are secure with probability at least q then n sessions of the composed protocol are secure with probability at least pq, provided the protocol messages are tagged with the information of which protocol they belong to. Indeed, a similar result is claimed in [18] for the non-randomized protocols. Unfortunately, this result is not valid even for nonrandomized protocols and we exhibit a simple example which contradicts this desired result (See

Example 5 in Appendix A). Essentially, the reason for this failure is that, in the claimed result, the n sessions of Q are assumed to generate fresh shared secrets in every session; but P may not be guaranteeing this freshness. Thus, messages of one session can get confused with messages of other sessions. We establish a weaker composition result in which we assume that the messages of each session of Q are tagged with a *unique* session identifier in addition to the protocol identifier. The use of session identifiers ensures that the messages of one session cannot be confused with other sessions.

Finally, we also consider the case for protocols containing an unbounded number of sessions. For this case, we observe that a composition result is only possible when P and Q are secure with probability with exactly 1. This is because if m sessions of a protocol leak a secret with probability $r > 0$ then by running mk sessions we can amplify the probability of leaking the secret. This probability approaches 1 as we increase k. We show that if an unbounded number of sessions of P are secure with probability 1 and an unbounded number of sessions of Q are secure with probability 1 then an unbounded number of sessions of the composed protocol are secure with probability 1, if the protocol messages are tagged with the information of which protocol they belong to and the messages of each session of Q are tagged with a *unique* session identifier.

The paper is organized as follows. In Section 2 we give relevant background information. Section 3 presents our processes algebra for randomized protocols and Section 4 gives our main composition result. In Section 5 we extend our results to tagged protocols and in Section 6 we show how to safely compose protocols with multiple sessions.

2 Preliminaries

We will start by discussing some standard notions from probability theory, Markov Chains and Markov Decision Processes. A process algebra for modeling security protocols with coin tosses will then be presented in Section 3. This process algebra closely resembles that of [25], which extends the applied π-calculus by the inclusion of a new operator for probabilistic choice. Following [18], our process calculus will also include several limitations necessary to achieve our results. In particular, conditionals no longer include else branches and we consider only a single public channel.

2.1 Probability spaces

We will assume the reader is familiar with probability spaces and give only the necessary definitions. A (sub)-probability space on S is a tuple $\Omega = (X, \Sigma, \mu)$ where Σ is a σ-algebra on X and $\mu : \Sigma \to [0, 1]$ is a countably additive function such that $\mu(\emptyset) = 0$ and $\mu(X) \leq 1$. The set Σ is said to be the set of events and μ the (sub)-probability measure of Ω. For $F \in \Sigma$, the quantity $\mu(F)$ is said to be the probability of the event F. If $\mu(X) = 1$ then we call μ a probability measure. Given two (sub)-probability measures μ_1 and μ_2 on a measure space

(S, Σ) as well as a real number $p \in [0,1]$, the convex combination $\mu_1 +_p \mu_2$ is the (sub)-probability measure μ such that for each set $F \in \Sigma$ we have $\mu(F) = p \cdot \mu_1(F) + (1-p) \cdot \mu_2(F)$. The set of all discrete probability distributions over S will be denoted by $\mathsf{Dist}(S)$. Given any $x \in S$, the *Dirac measure* on S, denoted δ_x, is the discrete probability measure μ such that $\mu(x) = 1$.

2.2 Discrete-time Markov Chains (DTMCs)

A DTMC is used to model systems which exhibit probabilistic behavior. Formally, a DTMC is a tuple $\mathcal{M} = (Z, z_s, \Delta)$ where Z is a countable set of *states*, z_s the *initial* state and $\Delta : Z \hookrightarrow \mathsf{Dist}(Z)$ is the (partial) *transition function* which maps Z to a (discrete) probability distribution over Z. Informally, the process modeled by $\mathcal{M}$ evolves as follows. The process starts in the state z_s. After i execution steps, if the process is in the state z, the process moves to state z' at execution step $(i+1)$ with probability $\Delta(z)(z')$.

An execution of $\mathcal{M}$ is a (finite or infinite) sequence $z_0 \to z_1 \to z_2 \cdots$ such that $z_0 = z_s$ and for each $i \geq 0$, $\Delta(z_i)(z_{i+1}) > 0$. The set of all executions of $\mathcal{M}$ will be denoted by $\mathsf{Exec}(\mathcal{M})$. The measure of finite sequence $z_0 \to z_1 \to z_2 \cdots \to z_m$ is defined to be $(\Delta(z_0)(z_1)) \cdot (\Delta(z_1)(z_2)) \cdot \ldots \cdot (\Delta(z_{m-1})(z_m))$. This extends to a unique (sub)-probability measure on the σ-algebra generated by $\mathsf{Exec}(\mathcal{M})$.

An execution ρ_1 is said to be a *one-step extension* of the execution $\rho = z_0 \to z_1 \to z_2 \cdots \to z_m$ if there exists z_{m+1} such that $\rho_1 = z_0 \to z_1 \to z_2 \cdots \to z_m \to z_{m+1}$. In this case, we say that ρ_1 extends ρ by z_{m+1}. For a finite execution ρ, as given above, we say $last(\rho) = z_m$. Notice that the set of all executions of $\mathcal{M}$ closed under one step extension forms a tree whose root node is z_s. Each node in this tree is an execution of $\mathcal{M}$.

We will often need to reason about a set of finite executions of $\mathcal{M}$. When interpreting the set of all executions of $\mathcal{M}$ as a rooted tree, this can be thought of as pruning each branch of the tree at a certain depth. We formalize this through the notion of a restriction. Given a prefix closed set of executions $\mathcal{R}$ for $\mathcal{M}$, $\mathcal{M}$ restricted to $\mathcal{R}$, denoted $\mathcal{M}|_{\mathcal{R}}$, is the tree whose nodes are the intersection of $\mathcal{R}$ and original nodes of the tree.

2.3 Partially Observable Markov Decision Processes (POMDP)s

POMDPs are used to model processes which exhibit both probabilistic and non-deterministic behavior, where the states of the system are only partially observable. Formally, an POMDP is a tuple $\mathcal{M} = (Z, z_s, Act, \Delta, \equiv)$ where Z is a countable set of *states*, $z_s \in Z$ is the *initial* state, Act is a (countable) set of *actions*, $\Delta : Z \times Act \hookrightarrow Dist(Z)$ is a partial function called the *probabilistic transition relation* and $\equiv$ is an equivalence relation on Z. As a matter of notation, we shall write $z \xrightarrow{\alpha} \mu$ whenever $\Delta(z, \alpha) = \mu$. A POMDP is like a Markov Chain except that at each state z, there is a choice amongst several possible probabilistic transitions. The choice of which probabilistic transition to *trigger* is resolved by an *adversary.* Informally, the process modeled by $\mathcal{M}$ evolves as follows. The process starts in the state z_s. After i execution steps, if the process is in the state z, then the adversary chooses an action α such that $z \xrightarrow{\alpha} \mu$ and the process moves to state z' at the $(i+1)$-st execution step with probability $\mu(z')$.

An *execution* ρ in the POMDP $\mathcal{M}$ is a (finite or infinite) sequence $z_0 \xrightarrow{\alpha_1} z_1 \xrightarrow{\alpha_2} z_2 \cdots$ such that $z_0 = z_s$ and for each $i \geq 0$, $z_i \xrightarrow{\alpha_{i+1}} \mu_{i+1}$ and $\mu_{i+1}(z_{i+1}) > 0$. The set of all finite executions of $\mathcal{M}$ will be denoted by $\mathsf{Exec}(\mathcal{M})$ and the set of all infinite executions will be denoted by $\mathsf{Exec}^{\infty}(\mathcal{M})$. If $\rho = z_0 \xrightarrow{\alpha_1} z_1 \xrightarrow{\alpha_2} z_2 \cdots \xrightarrow{\alpha_m} z_m$ is a finite execution then we write $last(\rho) = z_m$ and say the length of ρ, denoted $|\rho|$ is m. An execution ρ_1 is said to be a *one-step extension* of the execution $\rho = z_0 \xrightarrow{\alpha_1} z_1 \xrightarrow{\alpha_2} z_2 \cdots \xrightarrow{\alpha_m} z_m$ if there exists α_{m+1} and z_{m+1} such that $\rho_1 = z_0 \xrightarrow{\alpha_1} z_1 \xrightarrow{\alpha_2} z_2 \cdots \xrightarrow{\alpha_m} z_m \xrightarrow{\alpha_{m+1}} z_{m+1}$. In this case, we say that ρ_1 extends ρ by (α_{m+1}, z_{m+1}). An execution is called maximal if it is infinite or if it is finite and has no one-step extension. For an execution $\rho = z_0 \xrightarrow{\alpha_1} z_1 \xrightarrow{\alpha_2} z_2 \cdots \xrightarrow{\alpha_m} z_m$ we write $tr(\rho)$ to represent the *trace* of ρ, defined as the sequence $z_0/\!\equiv \xrightarrow{\alpha_1} z_1/\!\equiv \xrightarrow{\alpha_2} z_2/\!\equiv \cdots \xrightarrow{\alpha_m} z_m/\!\equiv$. The set of all traces is denoted $\mathsf{Trace}(\mathcal{M})$.

As discussed above, the choice of which transition to take in an execution is resolved by an adversary. Formally, an *adversary* $\mathcal{A} : \mathsf{Trace}(\mathcal{M}) \to Act$ is a partial function. For $t_1, t_2 \in \mathsf{Trace}(\mathcal{M})$, if $t_1 = t_2$ and $\mathcal{A}(t_1)$ is defined then $\mathcal{A}(t_2)$ is defined. An adversary $\mathcal{A}$ resolves all non-determinism and the resulting behavior can be described by a DTMC $\mathcal{M}^{\mathcal{A}} = (\mathsf{Exec}(\mathcal{M}), z_s, \Delta^{\mathcal{A}})$ where for each $\rho \in \mathsf{Exec}(\mathcal{M}), \Delta^{\mathcal{A}}(\rho)$ is the discrete probability distribution on $\mathsf{Exec}(\mathcal{M})$ such that $\Delta^{\mathcal{A}}(\rho)(\rho_1) = \Delta(z, \alpha)$ if there exists a state z and action α such that ρ_1 extends ρ by (α, z) and is 0 otherwise.

POMDPs and State-Based Safety properties Given a POMDP $\mathcal{M} = (Z, z_s, Act, \Delta, \equiv)$, a set $\Psi \subseteq Z$ is said to be a *state-based safety property.* We say $\mathcal{M}$ satisfies Ψ with probability $\geq p$ against the adversary $\mathcal{A}$ (written $\mathcal{M}^{\mathcal{A}} \models_p \Psi$) if the probability of the event $\{\kappa \in (\mathsf{Exec}(\mathcal{M}^{\mathcal{A}})) \mid \kappa \models \Psi \text{ and } \kappa \text{ is maximal}\}$ in the DTMC $\mathcal{M}^{\mathcal{A}}$ is $\geq p$. Here, $\kappa \models \Psi$ if $\kappa = z_0 \xrightarrow{\alpha_1} z_1 \xrightarrow{\alpha_2} z_2 \cdots \xrightarrow{\alpha_m} z_m$ is such that $z_j \in \Psi$ for all $0 \leq j \leq m$. We say that $\mathcal{M}$ satisfies Ψ with probability $\geq p$ (written $\mathcal{M} \models_p \Psi$) if for all adversaries $\mathcal{A}$, $\mathcal{M}^{\mathcal{A}} \models_p \Psi$.

Proposition 1. *Given a POMDP* $\mathcal{M} = (Z, z_s, Act, \Delta, Z_0, obs, \equiv)$ *and a* state-based safety property Ψ*, if* $\mathcal{M} \not\models_p \Psi$ *then there exists an adversary* $\mathcal{A}$ *and a prefix closed finite set of executions* $\mathcal{R} \in \mathsf{Exec}(\mathcal{M}^{\mathcal{A}})$ *such that* $\mathcal{M}^{\mathcal{A}}|_{\mathcal{R}} \not\models_p \Psi$*.*

2.4 Equational theories and frames

A signature $\mathcal{F}$ contains a finite set of function symbols, each with an associated arity. We assume a countably infinite set of special constant symbols $\mathcal{N}$, which we call names and use to represent data generated freshly during a protocol execution. Variable symbols are the union of two disjoint sets $\mathcal{X}$ and $\mathcal{X}_w$ which will be used as protocol and frame variables, respectively. It is required that variable symbols are disjoint from $\mathcal{F}$.

Terms are built by the application of function symbols to variables and terms in the standard way. Given a signature $\mathcal{F}$, we use $\mathcal{T}(\mathcal{F},\mathcal{X})$ to denote the set of terms built over $\mathcal{F}$ and $\mathcal{X}$. The set of variables occurring in a term is denoted by $vars(t)$. A ground term is one that contains no free variables and substitution of variables by terms σ is denoted by $\{x_1 \mapsto t_1, ..., x_k \mapsto t_k\}$. Given σ, its domain, denoted $dom(\sigma)$, is the set of variables $\{x_1, ..., x_k\}$ and its range, denoted $ran(\sigma)$, is the set of terms $\{t_1, .., t_k\}$. A substitution is said to be ground if every term in its range is ground.

Our process algebra is parameterized by a non-trivial equational theory $(\mathcal{F}, E)$, where E is a set of $\mathcal{F}$-Equations. By a $\mathcal{F}$-Equation, we mean a pair $l = r$ where $l, r \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}, \mathcal{X})$ are terms that do not contain names. Two terms s and t are said to be equal with respect to an equational theory $(\mathcal{F}, E)$, denoted $s =_E t$, if $E \vdash s = t$ in the first order theory of equality. For equational theories defined in the preceding manner, if two terms containing names are equivalent, they will remain equivalent when the names are replaced by arbitrary terms. We often identify an equational theory $(\mathcal{F}, E)$ by E when the signature is clear from the context. Processes are executed in an environment that consists of a frame φ and a binding substitution σ. Formally, $\sigma : \mathcal{X} \to \mathcal{T}(\mathcal{F})$ is a binding substitution and $\varphi : (\mathcal{X}_w \times \mathcal{E}) \to \mathcal{T}(\mathcal{F})$, where $\mathcal{E}$ is a set of equivalence classes, is called a frame. Two frames φ_1 and φ_2 are said to be statically equivalent if $dom(\varphi_1) = dom(\varphi_2)$ and for all $r_1, r_2 \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}, \mathcal{X}_w)$ we have $r_1\varphi_1 =_E r_2\varphi_1$ iff $r_1\varphi_2 =_E r_2\varphi_2$. Intuitively, two frames are statically equivalent if an attacker cannot distinguish between the information they contain.

Definition 1. *A term $t \in \mathcal{T}(\mathcal{F})$ is deducible from a frame φ with recipe $r \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}, dom(\varphi))$ in equational theory E, denoted $\varphi \vdash^r_E t$, if $r\varphi =_E t$. We often omit r and E and write $\varphi \vdash t$ if they are clear from the context.*

For the rest of the paper, $\mathcal{F}_b$ and $\mathcal{F}_c$ are signatures with disjoint sets of function symbols and $(\mathcal{F}_b, E_b)$ and $(\mathcal{F}_c, E_c)$ are non-trivial equational theories. The combination of these two theories will be $(\mathcal{F}, E) = (\mathcal{F}_b \cup \mathcal{F}_c, E_b \cup E_c)$.

3 Process Syntax and semantics

Our process syntax and semantics is similar to that of [18] with the addition of an operator for probabilistic choice. It can also been seen as a variant of [25].

Process Syntax: For technical reasons, we assume a countably infinite set of labels $\mathcal{L}$ and an equivalence relation $\sim$ on $\mathcal{L}$ that induces a countably infinite set of equivalence classes. For $l \in \mathcal{L}$, $[l]$ denotes the equivalence class of l. We use $\mathcal{L}_b$ and $\mathcal{L}_c$ to range over subsets of $\mathcal{L}$ such that $\mathcal{L}_b \cap \mathcal{L}_c = \emptyset$. If $l \in \mathcal{L}_b$ and $l' \in \mathcal{L}_c$ then $[l] \neq [l']$ and both $\mathcal{L}_b$ and $\mathcal{L}_c$ are closed under $\sim$. We assume each equivalence class contains a countably infinite number of labels. Each connective in our grammar will come with a label from $\mathcal{L}$, which will later be used to identify the process performing a protocol step after a composition takes place. The equivalence relation will be used to mask the information an adversary can obtain from the internal actions of a process, in the sense that, when an action with label l is executed, the adversary will only be able to infer $[l]$.

The syntax of processes is introduced in Figure 1. It begins by introducing what we call basic processes, which we will denote by $B, B_1, B_2, ...B_n$. In the definition of basic processes, $p \in [0, 1]$, $l \in \mathcal{L}$, $x \in \mathcal{X}$ and $c_i \in \{\top, s = t\} \forall i \in \{1, ..., k\}$ where $s, t \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}, \mathcal{X})$. In the case of the assignment rule $(x := t)^l$, we additionally require that $x \notin vars(t)$. Intuitively, basic processes will be used to represent the actions of a particular protocol participant. 0^l is a process that does nothing and νx^l is the process that creates a fresh name and binds it to x. The process $(x := t)^l$ assigns the term t to the variable x. The test process $[c_1 \wedge ... \wedge c_k]^l$ terminates if c_i is $\top$ or c_i is $s = t$ where $s =_E t$ for all $i \in \{1, ..., k\}$ and otherwise, if some c_i is $s = t$ and $s \neq_E t$, the process deadlocks. The process $in(x)^l$ reads a term t from the public channel and binds it to x and the process $out(t)^l$ outputs a term on the public channel. The processes $P \cdot^l Q$ sequentially executes P followed by Q whereas the process $P \oplus_p^l Q$ behaves like P with probability p and like Q with probability $1 - p$.

Basic Processes

$$B ::= 0^l \big| \nu x^l \big| (x := t)^l \big| [c_1 \wedge ... \wedge c_k]^l \big| in(x)^l \big| out(t)^l \big| (B \cdot^l B) \big| (B \oplus_p^l B)$$

Basic Contexts

$$D[\Box] ::= \Box \big| B \big| D[\Box] \cdot^l B \big| B \cdot^l D[\Box] \big| D[\Box] \oplus_p^l D[\Box]$$

Contexts $[a_i \in \{\nu x, (x := t)\}]$

$$C[\Box_1, ..., \Box_m] ::= a_1^{l_1} \cdot ... \cdot a_n^{l_n} \cdot (D_1[\Box_1] |^{l_{n+1}} D_2[\Box_2] |^{l_{n+2}} ... |^{l_{n+m-1}} D_m[\Box_m])$$

Fig. 1: Process Syntax

In Figure 1, basic processes are extended to include a special process variable $\Box$ and $\Box_1, ..., \Box_m$ are used to represent distinct processes variables. The resulting object is a basic context, which we will denote by $D[\Box]$, $D_1[\Box]$, $D_2[\Box]$, ..., $D_n[\Box]$. Notice that only a single process variable can appear in a basic context. $D_1[B_1]$ denotes the process that results from replacing every occurrence of $\Box$ in D_1 by B_1. A context is then a sequential composition of fresh variable creations and variable assignments followed by the parallel composition of a set of basic contexts. The prefix of variable creations and assignments is used to instantiate data common to one or more basic contexts. In the definition of contexts, $a \in \{\nu x, (x := t)\}$. A process is nothing but a context that does not contain any process variables. We will use $C, C_1, C_2, ..., C_n$ to denote contexts and P, Q or R to denote processes. For a context $C[\Box_1, ..., \Box_m]$ and basic processes $B_1, ..., B_m$, $C[B_1, ..., B_m]$ denotes the process that results from replacing the each process variable $\Box_i$ by B_i. In what follows, we define functions fv and bv that map basic processes to the set of variables that occur free and bound in the process, respectively.

$$bv(B), fv(B) = \begin{cases} \emptyset, vars(t) \cup vars(s) & \text{if } B \text{ is } [s = t] \\ \emptyset, vars(t) & \text{if } B \text{ is } out(t) \\ \{x\}, vars(t) & \text{if } B \text{ is } (x := t) \\ \{x\}, \emptyset & \text{if } B \in \{in(x), \nu(x)\} \\ bv(B_1) \cup bv(B_2), fv(B_1) \cup (fv(B_2) \setminus bv(B_1)) & \text{if } B = B_1 \cdot B_2 \\ bv(B_1) \cap bv(B_2), fv(B_1) \cup fv(B_2) & \text{if } B = B_1 \oplus_p B_2 \end{cases}$$

As usual, a process containing no free variables is called ground.

Definition 2. *A context $C[\Box_1, ..., \Box_m] = a_1 \cdot ... \cdot a_n \cdot (D_1[\Box_1]|...|D_m[\Box_m])$ is said to be well-formed if every operator has a unique label and for any labels l_1 and l_2 occurring in D_i and D_j for $i, j \in \{1, 2, ..., m\}$, if $i \neq j$ then $[l_1] \neq [l_2]$.*

For the remainder of this paper, contexts are assumed to be well-formed. A process that results from replacing process variables in a context by basic processes is also assumed to be well-formed.

Convention 1 *For readability, we will omit process labels when they are not relevant in a particular setting. Whenever new actions are added to a process, their labels are assumed to be fresh and not equivalent to any existing labels of that process.*

Process Semantics: Given a process P, an extended process is a 3-tuple (P, φ, σ) where φ is a frame and σ is a binding substitution. Semantically, a ground process P is a POMDP $[\![P]\!] = (Z, z_s, Act, \Delta, \equiv)$, where Z is the set of all extended processes, z_s is $(P, \emptyset, \emptyset)$, $Act = (\mathcal{T}(\mathcal{F} \setminus \mathcal{N}, \mathcal{X}_w) \cup \{\tau\}, \mathcal{L}/\sim)$ and Δ is a partial function from extended processes to Act. We now give some additional notation preceding our formal definitions of Δ and $\equiv$. Specifically, $\mu \cdot^l Q$ is the distribution μ_1 such that

$$\mu_1(P', \varphi, \sigma) = \begin{cases} \mu(P, \varphi, \sigma) \text{ if } P' = P \cdot^l Q \\ 0 \qquad\qquad\quad \text{otherwise} \end{cases}$$

The distributions $\mu|^l Q$ and $Q|^l \mu$ are defined similarly. We can now define $\Delta((P, \varphi, \sigma), \alpha) = \mu$ if $(P, \varphi, \sigma) \xrightarrow{\alpha} \mu$, as defined in Figure 2. Note that Δ is indeed well defined, as basic processes are deterministic and each equivalence class on $\mathcal{L}$ identifies a unique basic process. Given an extended process η, let $\overline{\eta}$ denote the set of all $(\S, [l])$ such that $(P, \varphi, \sigma) \xrightarrow{(\S,[l])} \mu$, $\S \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}, \mathcal{X}_w) \cup \{\tau\}$ and l is the label of an input or output action. Using this, we lift our notion of equivalence on frames from Section 2.4 to an equivalence $\equiv$ on extended processes by requiring that two extended processes $\eta = (P, \varphi, \sigma)$ and $\eta' = (P', \varphi', \sigma')$ are equivalent if $\overline{\eta} = \overline{\eta}'$ and $\varphi =_E \varphi'$.

INPUT
$$\frac{r \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}, \mathcal{X}) \quad \varphi \vdash^{r} t \quad x \notin dom(\sigma)}{(in(x)^{l}, \varphi, \sigma) \xrightarrow{(r,[l])} \delta_{(0,\varphi,\sigma\cup\{x\mapsto t\})}}$$

NEW
$$\frac{x \notin dom(\sigma) \quad n \text{ is a fresh name}}{(\nu x^{l}, \varphi, \sigma) \xrightarrow{(\tau,[l])} \delta_{(0,\varphi,\sigma\cup\{x\mapsto n\})}}$$

OUTPUT
$$\frac{vars(t) \subseteq dom(\sigma)}{(out(t)^{l}, \varphi, \sigma) \xrightarrow{(\tau,[l])} \delta_{(0,\varphi\cup\{w_{(|dom(\varphi)|+1,[l])}\mapsto t\sigma\},\sigma)}}$$

TEST
$$\frac{\forall i \in \{1,...,n\}, c_i \text{ is } \top \text{ or } c_i \text{ is } s = t \text{ where } vars(s,t) \subseteq dom(\sigma) \text{ and } s\sigma =_{E} t\sigma}{([c_1 \wedge ... \wedge c_n]^{l}, \varphi, \sigma) \xrightarrow{(\tau,[l])} \delta_{(0,\varphi,\sigma)}}$$

ASSIGN
$$\frac{vars(t) \subseteq dom(\sigma) \quad x \notin dom(\sigma)}{((x := t)^{l}, \varphi, \sigma) \xrightarrow{(\tau,[l])} \delta_{(0,\varphi,\sigma\cup\{x\mapsto t\sigma\})}}$$

NULL
$$\frac{(Q_0, \varphi, \sigma) \xrightarrow{\alpha} \mu}{(0 \cdot^{l} Q_0, \varphi, \sigma) \xrightarrow{\alpha} \mu}$$

SEQUENCE
$$\frac{Q_0 \neq 0 \quad (Q_0, \varphi, \sigma) \xrightarrow{\alpha} \mu}{(Q_0 \cdot^{l} Q_1, \varphi, \sigma) \xrightarrow{\alpha} \mu \cdot^{l} Q_1}$$

PBRANCH
$$\frac{}{(Q_1 \oplus_{p}^{l} Q_2, \varphi, \sigma) \xrightarrow{(\tau,[l])} \delta_{(Q_1,\varphi,\sigma)} +_{p} \delta_{(Q_2,\varphi,\sigma)}}$$

PARALLEL$_L$
$$\frac{(Q_0, \varphi, \sigma) \xrightarrow{\alpha} \mu}{(Q_0|^{l}Q_1, \varphi, \sigma) \xrightarrow{\alpha} \mu|^{l}Q_1}$$

PARALLEL$_R$
$$\frac{((Q_1, \varphi, \sigma) \xrightarrow{\alpha} \mu}{(Q_0|^{l}Q_1, \varphi, \sigma) \xrightarrow{\alpha} Q_0|^{l}\mu}$$

Fig. 2: Process semantics

Definition 3. *An extended process (Q, φ, σ) preserves the secrecy of $x \in vars(Q)$ in the equational theory $(\mathcal{F}, E)$, denoted $(Q, \varphi, \sigma) \models_{E} x$, if there is no $r \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}, dom(\varphi))$ such that $\varphi \vdash_{E}^{r} x\sigma$. We write* $\mathsf{Secret}(x)$*, for $x \in vars(Q)$, to represent the set of states of $[\![Q]\!]$ that preserve the secrecy of x. We also write* $\mathsf{Secret}(\{x_1, ..., x_n\})$ *to denote* $\mathsf{Secret}(x_1) \cap ... \cap \mathsf{Secret}(x_n)$*. We will often omit the braces $\{,\}$ for ease of notation.*

Notation 1 *Note that for process P and variables $x_1, ..., x_n \in vars(P)$,* $\mathsf{Secret}(\{x_1, ..., x_n\})$ *is a safety property of $[\![P]\!]$. We shall write* $P \models_{E,p} \mathsf{Secret}(\{x_1, ..., x_n\})$ *whenever* $[\![P]\!] \models_{p} \mathsf{Secret}(\{x_1, ..., x_n\})$.

4 Composition result for the finite disjoint case

We are now ready to present our first composition result. Our focus here will be on the scenario where two principals run a key establishment protocol over the signature $\mathcal{F}_c$ after which each principal uses the established secret to communicate in a protocol over the signature $\mathcal{F}_b$. Before formalizing this result, we show how a simple DC-net protocol using Diffie-Hellman (DH) for key exchange can be modeled in our composition framework. Using the results from Theorem 1, the security guarantees of each sub-protocol are achieved for the full protocol.

Example 1. In a simple DC-net protocol, two parties Alice and Bob want to anonymously publish two bits m_A and m_B, respectively. To achieve this, Alice and Bob agree on three private random bits k_0, k_1 and b and output a pair of messages according to the following scheme.

$$\begin{array}{lll}
\text{If } b=0 & \text{Alice: } M_{A,0} = k_0 \oplus m_A, & M_{A,1} = k_1 \\
 & \text{Bob: } \; M_{B,0} = k_0, & M_{B,1} = k_1 \oplus m_B \\
\text{If } b=1 & \text{Alice: } M_{A,0} = k_0, & M_{A,1} = k_1 \oplus m_A \\
 & \text{Bob: } \; M_{B,0} = k_0 \oplus m_B, & M_{B,1} = k_1
\end{array}$$

From the protocol output, the messages m_A and m_B can be retrieved as $M_{A,0} \oplus M_{B,0}$ and $M_{A,1} \oplus M_{B,1}$. The party to which the messages belong, however, remains unconditionally private, provided the exchanged secrets are not revealed. This protocol can be modeled using the following equational theory.

$$\begin{array}{lll}
\mathcal{F}_b = \{0, 1, \oplus, enc, dec, <:>, fst, snd\} & & \\
E_b = \{dec(enc(m,k),k) = m, & x \oplus 0 = x & x \oplus x = 0 \\
\quad x \oplus y = y \oplus x & (x \oplus y) \oplus z = x \oplus (y \oplus z) & \\
\quad fst(< x : y >) = x & snd(< x : y >) = y\} &
\end{array}$$

The role of Alice in this protocol is defined in our process syntax as

$$\begin{array}{ll}
A & = A_0 \cdot (m_A = 0 \oplus_{\frac{1}{2}} m_A = 1) \cdot \\
 & \quad (out(enc(0,k)) \cdot out(< k_0 \oplus m_A : k_1 >) \cdot A_1 \oplus_{\frac{1}{2}} \\
 & \quad out(enc(1,k)) \cdot out(< k_0 : k_1 \oplus m_A >) \cdot A_2) \\
A_0 & = (k_0 = 0 \oplus_{\frac{1}{2}} k_0 = 1) \cdot (k_1 = 0 \oplus_{\frac{1}{2}} k_1 = 1) \cdot out(enc(< k_0 : k_1 >, k))
\end{array}$$

where $A_i = in(z) \cdot [s = i] \cdot \nu s \cdot out(s)$, for $i \in \{1, 2\}$. The inclusion of A_1 and A_2 in Alice's specification are test the security of the protocol. The probability that an attacker can derive the secret s is exactly the probability an attacker can guess the message belonging to Alice after the protocol completes. We now give the specification of Bob's protocol $B_1 \mid B_2$ below.

$$\begin{array}{l}
B_0 = in(z_0) \cdot in(z_1) \cdot (k_0 = fst(dec(z_0, k))) \cdot \\
\quad (k_1 = snd(dec(z_1, k))) \cdot (b = dec(z_1, k)) \\
B_1 = B_0 \cdot (m_B = 0 \oplus_{\frac{1}{2}} m_B = 1) \cdot out(enc(m_B), k) \cdot [b = 0] \cdot out(< k_0 : k_1 \oplus m_B >) \\
B_2 = B_0 \cdot in(z_2) \cdot (m_B = dec(z_2, k)) \cdot [b = 1] \cdot out(< k_0 \oplus m_B : k_1 >)
\end{array}$$

Notice that the output of Bob depends on the value of Alice's coin flip. Because our process calculus does not contain else branches, the required functionality is simulated using the parallel and test operators. Our specification of the DC-net protocol requires Alice and Bob to have established a shared symmetric key k. A specification of the DH key exchange protocol to establish this key is given below. This process is parameterized by the signature $\mathcal{F}_c = \{g\}$ and equations $E_c = \{(g^x)^y = (g^y)^x\}$.

$$\begin{array}{l} C[\Box_0, \Box_1, \Box_2] = \nu y \cdot in(a) \cdot (A_k \cdot \Box_0 | (k := a^y) \cdot \Box_1 | (k := a^y) \cdot \Box_2 | out(g^y)) \\ A_k = \nu x \cdot out(g^x) \cdot in(b) \cdot (k := b^x) \end{array}$$

Now if $C[[\top], [\top], [\top]]$ preserves the secrecy of the shared key k and $\nu k \cdot (A|B_1|B_2)$ preserves the secrecy of k and s with probability at least $\frac{1}{2}$, then the composed protocol $C[A, B_1, B_2]$ preserves the secrecy of s with probability at least $\frac{1}{2}$. That is, if the DC-net specification does not reveal which message belongs to Alice, then neither does the DC-net protocol using DH key exchange to establish a secret communication channel between Alice and Bob.

The proof of Theorem 1 will utilize an extension of the seperation result from [18], which intuitively says that for a context C and a basic process B, if the composition $C[B]$, where C and B are over disjoint signatures and derive a set of variables with probability q, can be transformed into the composition $C'[B']$, where C' and B' represent α-renamings of C and B, such that $vars(C') \cap vars(B') = \emptyset$ and the same secret derivation guarantees are achieved. Our formulation of this result utilizes the following notation. Given a context C and a set of labels L, C^d_L is the context that results from replacing, in C, every variable $x \in vars(C)$ by x^d if the atomic process containing x has a label from L. Additionally, given a set of variables S, $S^d := \{x^d | x \in S\}$.

Lemma 1. *Let $C[\Box_1, ..., \Box_n]$ be a context over $\mathcal{F}_c$ with labels from $\mathcal{L}_c$ and $B_1, ..., B_n$ be basic processes over $\mathcal{F}_b$ with labels from $\mathcal{L}_b$. Further assume that $C[B_1, ..., B_n]$ is ground, $x_s \in \bigcup_{i=1}^n vars(B_i) \setminus vars(C)$ and $fv(B_i) \subseteq \{x_i\}$ for all $i \in \{1, ..., n\}$. For $q \in [0, 1]$ and $l_0, l_1, ..., l_n \in \mathcal{L}_b$, if $C[B_1, ..., B_n] \not\models_{E,q} \mathsf{Secret}(x_s)$ then $Q \not\models_{E,q} \mathsf{Secret}(S \cup S^b \cup x_s)$ where $S = \{x_1, ...x_n\}$ and*

$$Q := \nu k^{l_0} \cdot (x_1^b := k)^{l_1} \cdot ... \cdot (x_n^b := k)^{l_n} \cdot C[B_1, ..., B_n]^b_{\mathcal{L}_b}.$$

As a result of Lemma 1, an adversary for a composition of $C[B]$ can be transformed into an adversary for a composition of two protocols C' and B' over disjoint variables. From this adversary $\mathcal{A}$, we need to construct and an adversary $\mathcal{A}'$ for one of the sub-protocols C' or B'. Because C' and B' are simply α-renamings of C and B, $\mathcal{A}'$ is sufficient for contradicting a secrecy guarantee about one of the sub-protocols C or B. One of the challenges in constructing $\mathcal{A}'$ is that the adversary A may use terms over $\mathcal{F}_b \cup \mathcal{F}_c$. That is, it may construct inputs using terms output by both of the sub-protocols C' and B'. Our technique is to transform $\mathcal{A}$ into what we call a "pure" adversary, that constructs its inputs for actions of C' (resp B') using only terms output by actions of C' (resp B'). We define this concept formally. Given a set of labels L closed under $\sim$, let $\mathcal{X}^L_w = \{w_{i,[l]} \mid w_{i,[l]} \in \mathcal{X}_w \wedge l \in L \wedge i \in \mathbb{N}\}$

Definition 4. *Let L be a set of labels closed under $\sim$ and $\mathcal{F}$ be a signature. An adversary $\mathcal{A}$ for a process P is said to be pure with respect to $(L, \mathcal{F})$ if whenever $\mathcal{A}$ chooses the action $(r, [l])$ we have $r \in \mathcal{T}(\mathcal{F}, \mathcal{X}_w^L)$.*

In the following proposition, we prove formally that an adversary $\mathcal{A}$ for a composition of $C'[B']$ can be transformed into an adversary that is pure with respect to both sub-protocols.

Lemma 2. *Let $C[\Box_1, ..., \Box_n]$ be a context over $\mathcal{F}_c$ with variables from $\mathcal{X}_c$ and labels from $\mathcal{L}_c$. Likewise let $B_0, ..., B_n$ be basic processes over $\mathcal{F}_b$ with variables from $\mathcal{X}_b$ and labels from $\mathcal{L}_b$ where $\mathcal{X}_b$ and $\mathcal{X}_c$ are disjoint and $B_0 \cdot C[B_1, ..., B_n]$ is ground. If there exists an adversary $\mathcal{A}$ such that $[\![B_0 \cdot C[B_1, ..., B_n]]\!]^{\mathcal{A}} \not\models_{E,q}$ $\mathsf{Secret}(S)$ for $S \subseteq (vars(B) \cup vars(C))$ then there exists an adversary $\mathcal{A}'$ such that $[\![B_0 \cdot C[B_1, ..., B_n]]\!]^{\mathcal{A}'} \not\models_{E,q} \mathsf{Secret}(S)$ and $\mathcal{A}'$ is pure with respect to $(\mathcal{F}_b, \mathcal{L}_b)$ and $(\mathcal{F}_c, \mathcal{L}_c)$.*

We are now ready to give our main composition theorem. The fundamental technical challenge of this result is to show that, for some context $C[\Box]$, process B and set of secrets S, if $C[[\top]] \models_p \mathsf{Secret}(S)$ and $B \models_q \mathsf{Secret}(S)$, then $C[B] \models_{pq}$ $\mathsf{Secret}(S)$. In light of the preceding two results, this boils down to transforming a pure adversary for a composed protocol $C[B]$ that reveals some secret values with probability $\geq 1 - pq$ into an adversary for one of the sub-protocols C or B that derives the secret values with probability at least $1 - p$ or $1 - q$, respectively. Our technique is to first transform the adversary for $C[B]$ into an adversary for $C|B$. This adversary is then further transformed into one that runs C to completion before running the same (sub)-adversary for B in each execution path. The resulting adversary, in which the secret values are derived with probability $\geq 1 - pq$, allows one to extract an adversary for C or B that derives the secret values with the desired probability. To further understand these challenges, the interested reader should consult Example 4 in Appendix A.

Theorem 1. *Let $C[\Box_1, ..., \Box_n] = \nu k_1 \cdot ... \cdot \nu k_m \cdot (D_1[\Box_1] \mid D_2[\Box_2] \mid ... \mid D_n[\Box_n])$ be a context over $\mathcal{F}_c$ with labels from $\mathcal{L}_c$, $B_1, B_2, ..., B_n$ be basic processes over $\mathcal{F}_b$ with labels from $\mathcal{L}_b$, $q_1, q_2 \in [0, 1]$ and $x_s \in \bigcup_{i=1}^{n} vars(B_i) \setminus vars(C)$ such that:*

1. *$fv(C) = \emptyset$ and $fv(B_i) \subseteq \{x_i\}$*
2. *$vars(C) \cap vars(B_i) \subseteq \{x_i\}$ for $i \in \{1, ..., n\}$*
3. *$C[B_1, ..., B_n]$ is ground*
4. *$C[[\top]^{l_0}, ..., [\top]^{l_u}] \models_{E_c, q_1} \mathsf{Secret}(x_1, ..., x_n)$ where $l_0, ..., l_u \in \mathcal{L}_b$*
5. *$\nu k \cdot (x_1 := k) \cdot ... \cdot (x_n := k) \cdot (B_1 | ... | B_n) \models_{E_b, q_2} \mathsf{Secret}(x_1, ..., x_n, x_s)$*

Then $C[B_1, ..., B_n] \models_{E, q_1 q_2} \mathsf{Secret}(x_s)$.

As a result of Theorem 1, one can reason about protocols composed sequentially by taking a context with a single basic context where a single hole appears at the end. The same is true for protocols composed in parallel, as given by the following Corollary. In this setting, one considers a context built over two basic contexts. One basic context contains only a hole, while the other contains no holes.

Corollary 1. *Let C be a basic process over $\mathcal{F}_c$ with labels from $\mathcal{L}_c$ and B be a basic processes over $\mathcal{F}_b$ with labels from $\mathcal{L}_b$ and $q_1, q_2 \in [0,1]$ such that:*

1. $vars(C) \cap vars(B) = \emptyset$
2. $C \models_{E_c,q_1} \mathsf{Secret}(x_c)$ *for* $x_c \in vars(C)$
3. $B \models_{E_b,q_2} \mathsf{Secret}(x_b)$ *for* $x_b \in vars(B)$

Then $(C|B) \models_{E,q_1 q_2} \mathsf{Secret}(x_b, x_c)$.

It is important to point out that the security guarantees of the composed process may in fact be stronger than what we can prove utilizing Theorem 1. This is because we always assume the worst case in that context assigns the same secret values to each basic process. As a result, our composition result will in some cases lead to only an under-approximation on the probability that a set of variables is kept secret, as shown by the following example:

Example 2. Consider the signatures $\mathcal{F}_b = \{h\}$ and $\mathcal{F}_c = \{\}$ with empty equational theories and the context defined as follows:

$$C[\Box_1, \Box_2] = \nu k_1 \cdot \nu k_2 \cdot (([x_1 := k_1] \oplus_{\frac{1}{2}} [x_1 := k_2]) \cdot \Box_1 \mid [x_2 := k_2] \cdot \Box_2)$$

Essentially, the context generates shared secrets x_1 and x_2 for two sub-protocols $\Box_1$ and $\Box_2$ to be run in parallel. For the sub-protocol $\Box_1$, it sets the secret x_1 to k_1 with probability $\frac{1}{2}$ and to k_2 with probability $\frac{1}{2}$. In the second sub-protocol, the shared secret x_2 is set to k_2. Now consider the sub-protocols B_1 and B_2 defined as follows:

$$\begin{aligned} B_1 &= out(h(x_1)) \oplus_{\frac{1}{2}} 0 \\ B_2 &= in(z) \cdot [z = h(x_2)] \cdot \nu x_s \cdot out(x_s) \end{aligned}$$

B_1 outputs $h(x_1)$ with probability $\frac{1}{2}$ and with probability $\frac{1}{2}$ does nothing. B_2 checks if the adversary can construct $h(x_2)$ before revealing x_s. It is easy to see that $C[B_1, B_2]$ reveals x_s with probability $\frac{1}{4}$. This is because the adversary can construct $h(x_2)$ when x_1 and x_2 are equal (which happens with probability $\frac{1}{2}$) and when B_1 reveals $h(x_1)$ (which also happens with probability $\frac{1}{2}$). In-fact, we can easily show that $C[B_1, B_2]$ keeps x_s secret with probability exactly $\frac{3}{4}$. However, Theorem 1 can only show $C[B_1, B_2]$ keeps x_s secret with probability $\frac{1}{4}$, since in our composition results, we assume that x_1 and x_2 get the same secret name.

5 Achieving Disjointness Through Tagging

It is often necessary for protocols to share basic cryptographic primitives, such as functions for encryption, decryption and hashing. We extend our composition result to such kind of protocols. The key ingredient for composition in this context is tagging, a syntactic transformation of a protocol and its signature, designed to ensure secure composition. Essentially, tagging a protocol appends a special identifier to each of the messages that it outputs. When the protocol performs an input, it will recursively test all subterms in the input message to verify their tags are consistent with the protocol's tag. In the context of composition, tagging two protocols with different identifiers effectively achieves our disjointness condition from section 4. One limitation with tagging is that its correctness largely depends on the signature in question. As in [18], we will limit the class of cryptographic primitives we consider to symmetric encryption and a hash function, with the understanding that our results can be extended to primitives for asymmetric encryption. We now give the definition of tagging, as defined in [18], but repeated here for completeness.

Let C be a context and B be a basic process, both over the equational theory $(\mathcal{F}_{enc}, E_{enc})$ where $\mathcal{F}_{enc} = \{enc, dec, h\}$ and $E_{enc} = \{dec(enc(m, rn, k), k) = m\}$. To securely compose C and B, the terms occurring in each protocol must be tagged by function symbols from disjoint equational theories. In our setting, the tagging of two protocols will be done in two steps. To begin, the signature renaming function $\_^d$ from definition 5 will be applied to each of C and B with distinct values of $d \in \{b, c\}$.

Definition 5. *The signature renaming function $\_^d$ transforms a context C over the signature $(\mathcal{F}_{enc}, E_{enc})$ to a context C^d by replacing every occurrence of the function symbols enc, dec and h in C by enc_d, dec_d and h_d, respectively. The resulting context C^d is over the signature $(\mathcal{F}^d_{enc}, E^d_{enc})$, for $\mathcal{F}^d_{enc} = \{enc_d, dec_d, h_d\}$ and $E^d_{enc} = \{dec_d(enc_d(m, rn, k), k) = m\}$.*

Given C^c and B^b over the disjoint signatures $\mathcal{F}^c_{enc}$ and $\mathcal{F}^b_{enc}$, the function $\lceil \_ \rceil$ is then applied to C^c and B^b, generating the the tagged versions of C and B. We now give some prerequisite definitions for the definition of $\lceil \_ \rceil$. Let $\mathcal{F}^d_{tag} = \{tag_d, untag_d\}$ and $E^d_{tag} = \{untag_d(tag_d(x)) = x\}$. Further, $\mathcal{F}_{tag} = \mathcal{F}^b_{tag} \cup \mathcal{F}^c_{tag}$ and $E_{tag} = E^b_{tag} \cup E^c_{tag}$.

Definition 6. *Let $t \in \mathcal{F}^d_{enc}$ for $d \in \{b, c\}$. The function $\mathcal{H} : \mathcal{T}(\mathcal{F}^d_{enc}, \mathcal{X}) \to \mathcal{T}(\mathcal{F}_{enc} \cup \mathcal{F}^d_{tag}, \mathcal{X})$ is defined below.*

$$\begin{aligned}
\mathcal{H}(enc_d(t_1, t_2, t_3)) &= enc(tag_d(\mathcal{H}(t_1)), \mathcal{H}(t_2), \mathcal{H}(t_3)\\
\mathcal{H}(dec_d(t_1, t_2)) &= untag_d(dec(\mathcal{H}(t_1), \mathcal{H}(t_2)))\\
\mathcal{H}(h_d(t_1)) &= h(tag_d(\mathcal{H}(t_1)))\\
\mathcal{H}(u) &= u \quad \textit{if } u \textit{ is a name or variable}
\end{aligned}$$

The function $tests^d$ below maps terms from $\mathcal{T}(\mathcal{F}_{enc} \cup \mathcal{F}^d_{tag}, \mathcal{X})$ to a conjunction of equalities, as defined below.

$$\begin{aligned}
tests^d(enc(t_1, t_2, t_3)) &= tests^d(t_1) \wedge tests^d(t_2) \wedge tests^d(t_3)\\
tests^d(dec(t_1, t_2)) &= tests^d(t_1) \wedge tests^d(t_2)\\
tests^d(h(t_1)) &= tests^d(t_1)\\
tests^d(tag_d(t_1)) &= tests^d(t_1)\\
tests^d(untag_d(t_1)) &= tag_d(untag_d(t_1)) = t_1 \wedge\ tests^d(t_1)\\
tests^d(u) &= \top \quad \textit{if } u \textit{ is a name or variable}
\end{aligned}$$

We can now define $\lceil \_ \rceil$ as follows.

Definition 7. *Let C^d be a context over $\mathcal{F}^d_{enc}$ for $d \in \{b, c\}$. The context $\lceil C^d \rceil$ is defined as follows.*

$$\begin{array}{ll}
\lceil \Box \rceil & = \Box \\
\lceil \nu x \rceil & = [\top] \cdot \nu x \\
\lceil in(x) \rceil & = [\top] \cdot in(x) \\
\lceil out(t) \rceil & = [tests^d(\mathcal{H}(t))] \cdot out(\mathcal{H}(t)) \\
\lceil (x := t) \rceil & = [tests^d(\mathcal{H}(t))] \cdot (x := \mathcal{H}(t)) \\
\lceil [s = t] \rceil & = [tests^d(\mathcal{H}(s)) \wedge tests^d(\mathcal{H}(t))] \cdot [\mathcal{H}(s) = \mathcal{H}(t)] \\
\lceil C_1 \cdot C_2 \rceil & = \lceil C_1 \rceil \cdot \lceil C_2 \rceil \\
\lceil C_1 \oplus_p C_2 \rceil & = \lceil C_1 \rceil \oplus_p \lceil C_2 \rceil \\
\lceil C_1 | C_2 \rceil & = \lceil C_1 \rceil | \lceil C_2 \rceil
\end{array}$$

The following Proposition asserts the correctness of our tagging scheme. That is, whenever a protocol manipulates a term, that term should be tagged with the identifier of the protocol. To enforce this, every atomic action in a tagged protocol is prefixed with a conjunction of tests. If the terms manipulated by the atomic action meets the aforementioned requirement, the tests will pass. Otherwise, the tests will fail, and further protocol actions will be blocked. This is stated precisely as follows.

Proposition 2. *Let t be a ground term over $\mathcal{F}_{enc} \cup \mathcal{F}_{tag}$ and $d \in \{a, b\}$. Every $s \in st(t)$ of the form $untag_d(tag_{d'}(s))$ is such that $d = d'$ iff $tests^d(t) = c_1 \wedge \ldots \wedge c_n$ where c_i is $\top$ or $s_1 = s_2$ for ground terms $s_1, s_2 \in \mathcal{F}_{enc} \cup \mathcal{F}_{tag}$ such that $s_1 =_{E_{enc} \cup E_{tag}} s_2$.*

We have introduced tagging as a way to enforce the disjointness condition of section 4. In Proposition 3, we show that an attack on a composition of two tagged protocols originating from the same signature can be mapped to an attack on the composition of the protocols when the signatures are explicitly made disjoint. As a matter of notation, given a context $C[\Box_1, \ldots, \Box_n]$ and basic processes $B_1, \ldots, B_n$ we write $\lceil C[B_1, \ldots, B_n] \rceil$ to denote the process that results from plugging in the processes $\lceil B_1 \rceil, \ldots, \lceil B_n \rceil$ into the context $\lceil C[\Box_1, \ldots, \Box_n] \rceil$.

Proposition 3. *Let $C[\Box_1, \ldots, \Box_n]$ be a context over $\mathcal{F}_{enc}$ and $B_1, B_2, \ldots, B_n$ be basic processes over $\mathcal{F}_{enc}$. If $\lceil C^c[B^b_1, \ldots, B^b_n] \rceil \not\models_{E_{enc} \cup E_{tag}, q} \mathsf{Secret}(S)$ then $C^c[B^b_1, \ldots, B^b_n] \not\models_{E^b_{enc} \cup E^c_{enc}, q} \mathsf{Secret}(S)$ for $S \subseteq vars(C[B_1, \ldots, B_n])$ and $q \in [0, 1]$.*

Using Proposition 3 and Theorem 1, we can achieve the following result for tagged protocols.

Theorem 2. *Let $C[\Box_1, \ldots, \Box_n] = \nu k_1 \cdot \ldots \cdot \nu k_m \cdot (D_1[\Box_1] \mid D_2[\Box_2] \mid \ldots \mid D_n[\Box_n])$ be a context over $\mathcal{F}_{enc}$ with labels from $\mathcal{L}_c$, $B_1, B_2, \ldots, B_n$ be basic processes over $\mathcal{F}_{enc}$ with labels from $\mathcal{L}_b$, $q_1, q_2 \in [0, 1]$ and $x_s \in \bigcup_{i=1}^{n} vars(B_i) \setminus vars(C)$ such that:*

- *$fv(C) = \emptyset$ and $fv(B_i) \subseteq \{x_i\}$*

– $vars(C) \cap vars(B_i) \subseteq \{x_i\}$ *for* $i \in \{1, ..., n\}$
– $C[B_1, ..., B_n]$ *is ground*
– $C[0, ..., 0] \models_{E_{enc}, q_1} \mathsf{Secret}(x_1, ..., x_n)$
– $\nu k \cdot (x_1 := k) \cdot ... \cdot (x_n := k) \cdot (B_1 | ... | B_n) \models_{E_{enc}, q_2} \mathsf{Secret}(x_1, ..., x_n, x_s)$

Then $\lceil C^c[B_1^b, ..., B_n^b] \rceil \models_{E_{enc} \cup E_{tag}, q_1 q_2} \mathsf{Secret}(x_s)$.

6 Replication

In this section, we extend our composition result to protocols that can run multiple sessions.[3] We will begin by considering processes that contain only a bounded version of the replication operator. The bounded replication operator has an explicit bound that limits the number of times a process can replicate. As will be seen later, there are some challenges in dealing with replication in its full generality. We will also limit ourselves to processes that contain only a single occurrence of this replication operator. This restriction is not limiting for the applications we consider and it will simplify the proofs. It is, however, possible to extends our results to a more general setting in which a process can contain multiple occurrences of the replication operator.

We will start by showing showing that if the protocols $C = \nu k_1 \cdot ... \cdot \nu k_m \cdot !_n(C[\square_1] | ... | C[\square_l])$ and $!_n(B_1 | ... | B_l)$ are proven secure with probability at least p and q, respectively, then the composition $\nu k_1 \cdot ... \cdot \nu k_m \cdot !_n(C[B_1] | ... | C[B_l])$ is secure with probability at least pq, provided the protocol messages are tagged with both a protocol identifier and a unique session identifier. A similar result (with the absence of the session identifier), has been claimed in [18] for nonrandomized protocols (with p and q both being 1). However, we discovered a simple counterexample (Example 5 in Appendix A), which works for the case of two sessions. Essentially the reason for this attack is that protocol messages from one session of Q can be confused with messages from the other session.

6.1 Bounded Replication

Formally, a context containing bounded replication is defined as

$$C[\square_1, ..., \square_m] ::= a_1^{l_1} \cdot ... \cdot a_n^{l_n} \cdot !_n^l (D_1[\square_1] |^{l_{n+1}} D_2[\square_2] |^{l_{n+2}} ... |^{l_{n+m-1}} D_m[\square_m])$$

where $a \in \{\nu x, (x := t)\}$ and $n \geq 2$ is a natural number. The semantics for this bounded replication operator is given in Figure 3, where $i, j \in \mathbb{N}$ are used to denoted the smallest previously unused indices. We will use $P(i)$ to denote that process that results from renaming each occurrence of $x \in vars(P)$ to x^i for $i \in \mathbb{N}$. When $P(i)$ or $P(j)$ is relabeled freshly as in Figure 3, the new labels must all belong to the same equivalence class (that contains only those labels). The notation x^* denotes the infinite set $\{x^0, x^1, x^2, ...\}$.

[3] n sessions of P will be denoted by $!_n P$.

$$\text{B-REPLICATION} \quad \frac{n > 2 \quad l' \text{ is a fresh label} \quad P(i) \text{ is relabeled freshly}}{(!^l_n P, \varphi, \sigma) \xrightarrow{(\tau,[l])} \delta_{(P(i)|^{l'} !^l_{n-1} P, \varphi, \sigma)}}$$

$$\text{B-REPLICATION}_{n=2} \quad \frac{l' \text{ is a fresh label} \quad P(i), P(j) \text{ are relabeled freshly}}{(!^l_2 P, \varphi, \sigma) \xrightarrow{(\tau,[l])} \delta_{(P(i)|^{l'} P(j), \varphi, \sigma)}}$$

Fig. 3: Bounded Replication semantics

Our semantics imposes an explicit variable renaming with each application of a replication rule. The reason for this best illustrated through an example. Consider the process $!_m in(x) \cdot P$ and the execution

$$(!_m in(x) \cdot P, \emptyset, \emptyset) \to^* (in(x) \cdot P | !_{m-1} in(x) \cdot P, \varphi, \{x \mapsto t\} \cup \sigma)$$

where variable renaming does not occur. This execution corresponds to the adversary replicating $!_m in(x) \cdot P$, running one instance of $in(x) \cdot P$ and then replicating $!_m in(x) \cdot P$ again. Note that, because x is bound at the end of the above execution, the semantics of the input action cause the process to deadlock at $in(x)$. In other words, an adversary can only effective run one copy of $!_m in(x) \cdot P$ for any process of the form $!_m in(x) \cdot P$. It is also convenient to consider this restricted version of α-renaming in view of secrecy. In particular, if a variable is α-named arbitrarily with each application of "B-REPLICATION", then the definition of $!^l_n P$ keeping $x \in vars(P)$ secret becomes unclear, or at least more complicated.

As mentioned in Example 5, our composition result must prevent messages from one session of a process with bounded replication from being confused with messages from another sessions. We achieve this in the following way. Our composed processes will contain an occurrence of $\nu\lambda$ directly following the occurrence of a bounded replication operator. This freshly generated "session tag" will then be used to augment tags occurring in the composed processes. We have the following result.

Theorem 3. *Let $C[\Box_1, ..., \Box_n] = \nu k_1 \cdot ... \cdot \nu k_m \cdot !_u \nu\lambda \cdot (D_1[\Box_1] \mid D_2[\Box_2] \mid ... \mid D_n[\Box_n])$ be a context over $\mathcal{F}_{enc}$ with labels from $\mathcal{L}_c$, $B_1, B_2, ..., B_n$ be basic processes over $\mathcal{F}_{enc}$ with labels from $\mathcal{L}_b$, $q_1, q_2 \in [0, 1]$ and $x_s \in \bigcup_{i=1}^n vars(B_i) \setminus vars(C)$ such that:*

- $fv(C) = \emptyset$ *and* $fv(B_i) \subseteq \{x_i\}$
- $vars(C) \cap vars(B_i) \subseteq \{x_i\}$ *for* $i \in \{1, ..., n\}$
- $\lambda \notin vars(P) \cup vars(Q)$
- $C[B_1, ..., B_n]$ *is ground*
- $C[0, ..., 0] \models_{E_{enc}, q_1} \mathsf{Secret}(x_1, ..., x_n)$
- $\nu k \cdot (x_1 := k) \cdot ... \cdot (x_n := k) \cdot !_m (B_1 | ... | B_n) \models_{E_{enc}, q_2} \mathsf{Secret}(x_1, ..., x_n, x^*_s)$

Then $\lceil \nu k_1 \cdot ... \cdot \nu k_m \cdot !_u \nu\lambda \cdot (D^c_1[B^{(b,\lambda)}_1] \mid D^c_2[B^{(b,\lambda)}_2] \mid ... \mid D^c_n[B^{(b,\lambda)}_n]) \rceil \models_{E_{enc} \cup E_{tag}, q_1 q_2}$ $\mathsf{Secret}(x^*_s)$.

6.2 Unbounded Replication

As a final result, we will show how protocols containing unbounded replication can be composed. That is, we will consider processes over the following grammar.

$$C[\square_1, ..., \square_m] ::= a_1^{l_1} \cdot ... \cdot a_n^{l_n} \cdot !^l (D_1[\square_1]|^{l_{n+1}} D_2[\square_2]|^{l_{n+2}} ... |^{l_{n+m-1}} D_m[\square_m])$$

where $a \in \{\nu x, (x := t)\}$. The semantics of this new replication operator are given in Figure 4, where again, $i \in \mathbb{N}$ is the smallest previously unused index. As before, when $P(i)$ is relabeled freshly, the new labels must all belong to the same equivalence class.

$$\text{B-REPLICATION}_N \qquad \frac{l' \text{ is a fresh label} \quad P(i) \text{ is relabeled freshly}}{(!^l P, \varphi, \sigma) \xrightarrow{(r,l)} \delta_{(P(i)|^{l'} !^l P, \varphi, \sigma)}}$$

Fig. 4: Replication semantics

As previously eluded to, it is difficult to state a result in the style of Theorem 3 with non-trivial probabilities.

$$\begin{tikzcd}
 & & & & (!P, \emptyset, \sigma) \arrow[dll, "\frac{1}{2}"'] \arrow[drr, "\frac{1}{2}"] & & & & \\
 & & (!P, \varphi_1, \emptyset) \arrow[dl, "\frac{1}{2}"'] \arrow[dr, "\frac{1}{2}"] & & & & (!P, \emptyset, \emptyset) \arrow[dl, "\frac{1}{2}"'] \arrow[dr, "\frac{1}{2}"] & & \\
 & (!P, \varphi_{12}, \emptyset) \arrow[dl, dotted] \arrow[dr, dotted] & & (!P, \varphi_1, \emptyset) \arrow[dl, dotted] \arrow[dr, dotted] & & (!P, \varphi_2, \emptyset) \arrow[dl, dotted] \arrow[dr, dotted] & & (!P, \emptyset, \emptyset) \arrow[dl, dotted] \arrow[dr, dotted] & \\
 {} & & {} & & {} & & {} & & {}
\end{tikzcd}$$

Fig. 5: Execution of $!P$.

Example 3. Consider the process $P = \nu x_s \cdot out(x_s) \oplus_{\frac{1}{2}} 0$ over the empty signature and equational theory. We have that $P \models_{E,\frac{1}{2}} \mathsf{Secret}(x_s)$. However, there is no $q \in (0, 1]$ such that $!P \models_{E,q} \mathsf{Secret}(x_s^*)$. To see this, let

$$\begin{aligned}
\varphi_0 \;&= \{x_s^0 \mapsto n_0, w_{l_0} \mapsto x_s^0\} \\
\varphi_{01} &= \{x_s^0 \mapsto n_0, x_s^1 \mapsto n_1, w_{l_0} \mapsto, x_s^0, w_{l_1} \mapsto x_s^1\} \\
\varphi_1 \;&= \{x_s^1 \mapsto n_1, w_{l_1} \mapsto x_s^1\}
\end{aligned}$$

and consider the execution in Figure 5. Each level n of the tree represents a finite execution in which n replications have occurred. Notice that an adversary choosing to replicate $!P$ a single time causes x_s^* to be revealed with probability $\frac{1}{2}$ and an adversary choosing to replication $!P$ twice causes x_s^* to be revealed with probability$\frac{3}{4}$. In general, the probability of revealing x_s^* tends towards 1 as the number of replications increases.

In light of example 3, our composition result must require secrets to be preserved with probability 1. Such a restriction makes the statement of our Theorem almost identical to that of Theorem 6 from [18]. Our result, however, has two main advantages. It elimiates the still applicable attack of Example 5 while considering a richer class of processes. We now have the following helper result, which is a consequence of Proposition 1.

Proposition 4. *If P is a process containing a single occurrence of $!$ such that $P \not\models_{E,1} \mathsf{Secret}(x \cup x^*)$, then there exists an n such that if P' is the result of replacing $!$ in P by $!_n$, then $P' \not\models_{E,1} \mathsf{Secret}(x \cup x^*)$, where $x \in vars(P)$.*

Using Proposition 4 and Theorem 2, we can prove the following.

Theorem 4. *Let $C[\Box_1, ..., \Box_n] = \nu k_1 \cdot ... \cdot \nu k_m \cdot !\nu\lambda \cdot (D_1[\Box_1] \mid D_2[\Box_2] \mid ... \mid D_n[\Box_n])$ be a context over $\mathcal{F}_{enc}$ with labels from $\mathcal{L}_c$, $B_1, B_2, ..., B_n$ be basic processes over $\mathcal{F}_{enc}$ with labels from $\mathcal{L}_b$ and $x_s \in \bigcup_{i=1}^n vars(B_i) \setminus vars(C)$ such that:*

- $fv(C) = \emptyset$ *and* $fv(B_i) \subseteq \{x_i\}$
- $vars(C) \cap vars(B_i) \subseteq \{x_i\}$ *for* $i \in \{1, ..., n\}$
- $\lambda \notin vars(P) \cup vars(Q)$
- $C[B_1, ..., B_n]$ *is ground*
- $C[0, ..., 0] \models_{E_{enc},1} \mathsf{Secret}(x_1, ..., x_n)$
- $\nu k \cdot (x_1 := k) \cdot ... \cdot (x_n := k) \cdot !_m(B_1 | ... | B_n) \models_{E_{enc},1} \mathsf{Secret}(x_1, ..., x_n, x_s^*)$

Then $\lceil \nu k_1 \cdot ... \cdot \nu k_m \cdot !\nu\lambda \cdot (D_1^c[B_1^{(b,\lambda)}] \mid D_2^c[B_2^{(b,\lambda)}] \mid ... \mid D_n^c[B_n^{(b,\lambda)}]) \rceil \models_{E_{enc} \cup E_{tag},1}$ $\mathsf{Secret}(x_s^*)$.

7 Conclusions

We have studied the problem of securely composing two randomized security protocols. For one session, we show that if P is secure with probability p and Q is secure with probability q then the composed protocol is secure with probability at least pq if the protocol messages are tagged with the information of which protocol they belong to. The same result applies to multiple sessions except that in addition the protocol messages of Q also need to be tagged with session identifiers. The focus of this work has been secrecy properties. In terms of future work, we plan to investigate when composition of randomized security protocols preserve indistinguishability properties.

References

1. Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *28th ACM Symp. on Principles of Programming Languages (POPL'01)*, pages 104–115, 2001.
2. Suzana Andova, Cas J. F. Cremers, Kristian Gjøsteen, Sjouke Mauw, Stig Fr. Mjølsnes, and Sasa Radomirovic. A framework for compositional verification of security protocols. *Information and Computation*, 206(2-4):425–459, 2008.
3. Myrto Arapinis, Vincent Cheval, and Stéphanie Delaune. Composing security protocols: from confidentiality to privacy. *preprint avaliable at: http://arxiv.org/pdf/1407.5444v3.pdf.*
4. Myrto Arapinis, Vincent Cheval, and Stéphanie Delaune. Verifying privacy-type properties in a modular way. In *25th IEEE Computer Security Foundations Symposium (CSF'12)*, pages 95–109, Cambridge Massachusetts, USA, 2012. IEEE Computer Society Press.
5. Myrto Arapinis, Stéphanie Delaune, and Steve Kremer. From one session to many: Dynamic tags for security protocols. In *15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'08)*, volume 5330 of *Lecture Notes in Artificial Intelligence*, pages 128–142. Springer, 2008.
6. Michael Ben-Or, Oded Goldreich, Silvio Micali, and Ronald L. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, 1990.
7. R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, P. Pereira, and R. Segala. Task-Structured Probabilistic I/O Automata. In *Workshop on Discrete Event Systems*, 2006.
8. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In Moni Naor, editor, *42nd IEEE Symp. on Foundations of Computer Science (FOCS'01)*, pages 136–145. IEEE Comp. Soc. Press, 2001.
9. Ran Canetti and Jonathan Herzog. Universally composable symbolic analysis of mutual authentication and key-exchange protocols (extended abstract). In *Proc. 3rd Theory of Cryptography Conference (TCC'06)*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403. Springer, 2006.
10. Marco Carbone and Joshua D. Guttman. Sessions and separability in security protocols. In *Proceedings of the Principles of Security and Trust - Second International Conference, POST 2013*, pages 267–286, 2013.
11. R. Chadha, A.P. Sistla, and M. Viswanathan. Model checking concurrent programs with nondeterminism and randomization. In *the International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 364–375, 2010.
12. K. Chatzikokolakis and C. Palamidessi. Making Random Choices Invisible to the Scheduler. *Information and Computation*, 2010, to appear.
13. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988.
14. L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices.* PhD thesis, Radboud University of Nijmegen, 2006.
15. Céline Chevalier, Stéphanie Delaune, and Steve Kremer. Transforming password protocols to compose. In *31st Conference on Foundations of Software Technology and Theoretical Computer Science*, Leibniz International Proceedings in Informatics, pages 204–216. Leibniz-Zentrum für Informatik, 2011.

16. Véronique Cortier, Jérémie Delaitre, and Stéphanie Delaune. Safely composing security protocols. In V. Arvind and Sanjiva Prasad, editors, *Proc. 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *Lecture Notes in Computer Science*, pages 352–363. Springer, December 2007.
17. Véronique Cortier and Stéphanie Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1):1–36, February 2009.
18. Ştefan Ciobâcă and Véronique Cortier. Protocol composition for arbitrary primitives. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010, Edinburgh, United Kingdom, July 17-19, 2010*, pages 322–336, 2010.
19. Anupam Datta, Ante Derek, John C. Mitchell, and Dusko Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 13(3):423–482, 2005.
20. L. de Alfaro. The Verification of Probabilistic Systems under Memoryless Partial Information Policies is Hard. In *PROBMIV*, 1999.
21. Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Composition of password-based protocols. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 239–251. IEEE Computer Society Press, June 2008.
22. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.
23. F.D. Garcia, P. van Rossum, and A. Sokolova. Probabilistic Anonymity and Admissible Schedulers. *CoRR*, abs/0706.1019, 2007.
24. David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Onion routing. *Commun. ACM*, 42(2):39–41, 1999.
25. Jean Goubault-Larrecq, Catuscia Palamidessi, and Angelo Troina. A probabilistic applied pi-calculus. In *5th Asian Symposium on Programming Languages and Systems (APLAS'07)*, volume 4807 of *Lecture Notes in Computer Science*, pages 175–290. Springer, 2007.
26. Carl A. Gunter, Sanjeev Khanna, Kaijun Tan, and Santosh S. Venkatesh. Dos protection for reliably authenticated broadcast. In *NDSS*, 2004.
27. Joshua D. Guttman. Authentication tests and disjoint encryption: A design method for security protocols. *Journal of Computer Security*, 12(3-4):409–433, 2004.
28. Joshua D. Guttman. Cryptographic protocol composition via the authentication tests. In Luca de Alfaro, editor, *the Foundations of Software Science and Computational Structures, 12th International Conference (FOSSACS 2009)*, volume 5504 of *LNCS*, pages 303–317. Springer, 2009.
29. Changhua He, Mukund Sundararajan, Anupam Datta, Ante Derek, and John C. Mitchell. A modular correctness proof of ieee 802.11i and TLS. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *the 12th ACM Conference on Computer and Communications Security, (CCS 2005)*, pages 2–15. ACM, 2005.
30. Sebastian Mödersheim and Luca Viganò. Sufficient conditions for vertical composition of security protocols. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '14, pages 435–446, New York, NY, USA, 2014. ACM.
31. Michael O. Rabin. Transaction protection by beacons. *J. Comput. Syst. Sci.*, 27(2):256–267, 1983.
32. Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998.
33. P. Y. A. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia. Prêt à voter: a voter-verifiable voting system. *IEEE Transactions on Information Forensics and Security*, 4(4):662–673, 2009.

A Examples

The examples here are intended to highlight the main challenges a) that arise in the context of *probabilistic* protocols and b) that arise in the context of multiple sessions. We start by giving an overview of the proof of the composition result in [18] for one session. For this section, we shall consider the composition of processes P and Q where P and Q are processes over disjoint signatures $\mathcal{F}_a$ and $\mathcal{F}_b$. When the signatures are not disjoint, we can tag the messages to achieve a sufficient disjointness condition. In a typical scenario discussed in [18], P represents a key establishment between two principals, where P is modeled as a parallel composition of P_1 and P_2 which represent the two principals. At the end of the protocol, the first principal has a shared key stored in the variable x_k and the second principal has a shared key shared in the variable y_k (note that these keys can be arbitrary terms and not just a secret name). The established secret key(s) can then be used by the two principals in the protocol Q to achieve some security goal. Q itself is a composition of Q_1 and Q_2 representing the two principals, where x_k and y_k are free in Q_1 and Q_2, respectively. It is then shown that the protocol $W = P_1 \cdot Q_1 | P_2 \cdot Q_2$ preserves a secret x_s of Q if (a) P keeps the shared keys secret and (b) assuming that x_k and y_k are given the *same secret name* at the beginning of the execution of Q, Q also keeps the shared keys secret. The proof is carried as follows:

- Note the values of x_k and y_k given by P could be related arbitrarily. However, since the process calculus does not contain inequality tests and the signatures $\mathcal{F}_a$ and $\mathcal{F}_b$ are disjoint, we can show that Q also preserves secrecy of x_s even when x_k and y_k are given distinct values.
- The proof now proceeds by contradiction. Assuming W reveals x_s, there must be an execution ρ of W that reveals x_s.
- ρ can then be viewed as an interleaving of *one* symbolic trace of P and *one* symbolic trace of Q. This, combined with the fact that P are Q are processes over disjoint signatures, allows ρ to be viewed as an interleaving of *one* execution of P and *one* execution of Q in which the messages of the alien processes are mapped to fresh names.
- Since P and Q execute independently of each other in the interleaving, it then follows that either P reveals one of the shared secrets or Q reveals x_s or one of the shared secrets.

In the case where P and/or Q are randomized protocols, we want to claim that if P is secure with probability at least p and Q is secure with probability at least q, then the composition is secure with probability at least pq. We will follow the same broad plan as in [18], but utilize different proof techniques to handle the issues specific to randomized protocols. In our context, an execution of a protocol is no longer a trace, but rather, a tree. This implies that when a random choice is made, the protocols may evolve differently in different branches of the tree. In particular, we will no longer be able to write an execution of W as an interleaving of one execution of P and one execution of Q. Additionally, recall that we are considering a weaker class of adversaries than [18], to account for protocols being able to make private coin tosses. In particular, we require that when two execution paths of a protocol reveal the same information information to the adversary, the adversary must take the same next action in both executions. As such, when mapping an adversary for a composed process to one of its sub-protocols, our techniques must preserve membership in this sub-class of adversaries. We illustrate these issues with an example.

Note that when we write (P, φ, σ) in the following examples, we mean the process P with the frame φ (which records the messages received by P thus far) and binding substitution σ (which assigns ground terms to variables in P). We shall use often labels on the actions to identify the processes they belong to.

Example 4. Consider the signatures $\mathcal{F}_a = \{c\}$ and $\mathcal{F}_b = \{h\}$ where c is a constant and h is a 1-ary function symbol. Let $E_a = E_b = \emptyset$ and P be the process defined as follows:

$$\begin{aligned} P\ &= P_1|P_2 \\ P_1 &= \nu x_k \cdot (out^1(x_k) \oplus_{\frac{1}{2}} out^2(c)) \\ P_2 &= \nu y_k \cdot (out^3(y_k) \oplus_{\frac{1}{2}} out^4(c)) \end{aligned}$$

Essentially P_1 generates x_k and with probability $\frac{1}{2}$ decides to reveal it. P_2 generates y_k and with probability $\frac{1}{2}$ decides to reveal it. In both cases, when the fresh values is not revealed, a constant is output in its place. Consider the process Q defined as follows

$$\begin{aligned} Q\ &= Q_1|Q_2 \\ Q_1 &= in^5(x) \cdot [x = x_k] \cdot \nu x_s \cdot out^6(x_s) \\ Q_2 &= in^7(y) \cdot [y = h(y_k)] \cdot \nu x_s \cdot out^8(x_s) \end{aligned}$$

Consider the process $W = P_1 \cdot Q_1 | P_2 \cdot Q_2$ and let $P_1', P_2', \sigma, , \varphi_1, \varphi_2, \varphi_{12}, \sigma^f, \varphi_1^f, \varphi_2^f$ and φ_{12}^f be defined as follows:

$$\begin{aligned} P_1' &= out^1(x_k) \oplus_{\frac{1}{2}} out^2(c) \\ P_2' &= out^3(y_k) \oplus_{\frac{1}{2}} out^4(c) \\ \sigma &= \{x_k \mapsto k_1, y_k \mapsto k_2\} \\ \varphi_0 &= \{w_1 \to c\} \\ \varphi_1 &= \{w_1 \to k_1\} \\ \varphi_2 &= \{w_1 \to c, w_2 \to k_2\} \\ \varphi_{00} &= \{w_1 \to c, w_2 \to c\} \\ \varphi_{10} &= \{w_1 \to k_1, w_2 \to c\} \\ \varphi_{12} &= \{w_1 \to k_1, w_2 \to k_2\} \\ \varphi_{02} &= \{w_1 \to k_1, w_2 \to c\} \\ \sigma_1^f &= \{x_k \mapsto k_1, y_k \mapsto k_2, x \mapsto k_1, x_s \mapsto k_3\} \\ \sigma_2^f &= \{x_k \mapsto k_1, y_k \mapsto k_2, y \mapsto h(k_1), x_s \mapsto k_3\} \\ \varphi_1^f &= \{w_1 \to k_1, w_2 \to c, w_4 \mapsto k_3\} \\ \varphi_2^f &= \{w_1 \to c, w_2 \to k_2, w_6 \mapsto k_3\} \\ \varphi_{12}^f &= \{w_1 \to k_1, w_2 \to k_2, w_4 \mapsto k_3\} \end{aligned}$$

The the execution of W shown in Figure 6 reveals x_s with probability $\frac{3}{4}$. Observe that the transitions out of the states labeling $(Q_1|Q_2, \varphi_1, \sigma)$ involve transitions of Q_1 while the transitions out of $(Q_1|Q_2, \varphi_2, \sigma)$ involve transitions of Q_2. If we try to fire the same transitions out of $(Q_1|Q_2, \varphi_2, \sigma)$ as in $(Q_1|Q_2, \varphi_1, \sigma)$ the process will deadlock because the adversary cannot deduce x_k in φ_2. From this, it is easy to see that the execution shown in Figure 6 cannot be written as an interleaving of one execution of P and one execution of Q. Nevertheless, we will be able to show that W keeps x_s secret with probability at least $\frac{1}{4}$.

In the execution of W shown in Figure 6, the adversary performs different actions depending of the result of coin toss made by P_1. When P_1 outputs a nonce, Q_1 is scheduled before Q_2. When P_1 outputs the constant c, Q_2 is executed first. Such an attack is valid, even when considering our restricted class of adversaries. The reason is that the adversary can infer the result of the coin toss in P_1 by observing what is output. However, consider the following alternate version of P given below.

$$\begin{aligned}
P'' &= P''_1|P''_2\\
P''_1 &= \nu x_k \cdot \nu n_1 \cdot (out^1(x_k) \oplus_{\frac{1}{2}} out^2(n_1))\\
P''_2 &= \nu y_k \cdot \nu n_2 \cdot (out^3(y_k) \oplus_{\frac{1}{2}} out^4(n_2))
\end{aligned}$$

Now the process $W' = P''_1 \cdot Q_1|P''_2 \cdot Q_2$ keeps x_s secret with probability at least $\frac{1}{2}$. This is because the adversary can no longer infer the result of the coin tosses from P, and is therefore required to act in a uniform way when scheduling Q_1 and Q_2. Our composition result correctly proves that W keeps x_s secret with probability at least $\frac{1}{4}$ and W' keeps x_s secret with probability at least $\frac{1}{2}$.

$$\begin{tikzcd}
 & & & (W, \emptyset, \emptyset) \arrow[d, "{\tau^*, 1}"'] & & & \\
 & & & (P'_1.Q_1|P'_2.Q_2, \emptyset, \sigma) \arrow[dll, "{out^1, \frac{1}{2}}"'] \arrow[drr, "{\tau^*, \frac{1}{2}}"] & & & \\
 & (Q_1|P'_2.Q_2, \varphi_1, \sigma) \arrow[dl, "{out^3, \frac{1}{2}}"'] \arrow[dr, "{\tau^*, \frac{1}{2}}"] & & & & (Q_1|P'_2.Q_2, \varphi_0, \sigma) \arrow[dl, "{out^3, \frac{1}{2}}"'] \arrow[dr, "{\tau^*, \frac{1}{2}}"] & \\
(Q_1|Q_2, \varphi_{12}, \sigma) \arrow[d, dotted, "{in^5.out^6, 1}"'] & & (Q_1|Q_2, \varphi_{10}, \sigma) \arrow[d, dotted, "{in^5.out^6, 1}"'] & & (Q_1|Q_2, \varphi_{02}, \sigma) \arrow[d, dotted, "{in^7.out^8, 1}"'] & & (Q_1|Q_2, \varphi_{00}, \sigma) \\
(\_, \varphi^f_{12}, \sigma^f_1) & & (\_, \varphi^f_1, \sigma^f_1) & & (\_, \varphi^f_2, \sigma^f_2) & &
\end{tikzcd}$$

Fig. 6: Execution of W. The solid edges are transition in P and dotted edges are transitions in Q. For convenience, the edges in the drawn execution tree may compose of more than 1 action. The recipes used in in^3 and in^5 are w_1 and $h(w_2)$ respectively. The transition probabilities also label the edges.

We would also like to extend the composition result to multiple sessions of protocols.[4] That is if the protocols $P = \nu k_1 \cdot \nu k_2 \cdot !_n(P_1|P_2)$ and $!_n Q$ are proven secure with probability at least p and q, respectively, then we would like to guarantee $W = \nu k_1 \cdot \nu k_2 \cdot !_n(P_1 \cdot Q_1 | P_2 \cdot Q_2)$ is secure with probability at least pq. Such a result has been claimed in [18] for nonrandomized protocols (with p and q being both 1). However, we discovered the following simple counterexample, which works for the case of two sessions. Essentially the reason for this attack is that protocol messages from one session of Q can be confused with messages from the other session.

Example 5. Consider the signatures $\mathcal{F}_b = \{h, c\}$ and $\mathcal{F}_a = \{\}$ where c is a constant, h is a 1-ary function symbol and $E = E_a \cup E_b = \emptyset$. We will consider two sessions of the composed protocol.

Let P be the process defined as:

$$P = \nu k_1 \cdot \nu k_2 \cdot !_2(P_1 \mid P_2)$$

where $P_1 = (x_k := k_1)$ and $P_2 = (y_k := k_2)$). Let Q be the process defined as:

$$\begin{aligned}
Q \; &= !_2(\nu k \cdot ((x_k := k) \cdot Q_1 \mid [y_k = k] \cdot Q_2)) \\
Q_1 &= (in(y) \cdot ([y = c] \cdot out^l(h(x_k)) \mid \\
&\qquad [y = h(x_k)] \cdot \nu x_s \cdot out^{l'}(x_s)) \\
Q_2 &= 0.
\end{aligned}$$

Clearly, P keeps x_k and y_k secret with probability 1 and Q keeps x_k, y_k and x_s secret with probability 1. Theorem 2 from [18] would imply that x_s is kept secret by W in both sessions of the protocol. However, we can show that this is not the case. The reason is as follows. In both sessions of the composed protocol, x_k gets the same value. In the first session of the composed protocol, when y is input by Q_1, attacker sends the constant c. Thereafter, the attacker learns $h(x_k)$ because Q_1 outputs it. In the second session of the composed protocol, the attacker sends $h(x_k)$ to Q_1; the check $[y_k = k]$ succeeds and the attacker learns x_s in this session.

In process calculus terms, this attack can be realized by the execution:

$$(W, \emptyset, \emptyset) \to^* (W', \emptyset, \sigma') \to^* (W'', \varphi'', \sigma'')$$

where

$$\begin{aligned}
W' \; &= (Q_1^1 | Q_1^2) \\
\sigma' \; &= \{k_1 \mapsto n_1, k_2 \mapsto n_2, x_k^1 \mapsto n_1, y_k^1 \mapsto n_2, \\
&\qquad x_k^2 \mapsto n_1, y_k^2 \mapsto n_2\} \\
W'' &= 0 \\
\sigma'' \; &= \sigma' \cup \{y^1 \mapsto c, y^2 \mapsto h(n_1), x_s^2 \mapsto n_3\} \\
\varphi'' \; &= \{w_l \mapsto h(n_1), w_{l'} \mapsto n_3\}
\end{aligned}$$

Note above that we have used superscripts on variables x_k, y_k, y and x_s in the substitutions to indicate their values in different sessions. Essentially in this

[4] n sessions of P will be denoted by $!_n P$.

execution in $(W', \emptyset, \sigma')$, P is finished in both sessions and assigned x_k and y_k the same values in both sessions. The role Q_2 is also finished in both sessions. Q_1^1 is the first session of Q_1 and Q_1^2 is the second session of Q_1. Now in Q_1^1, the adversary inputs c for y resulting in Q_1^1 leaking $h(n_1)$. In Q_1^2, the adversary can input $h(n_1)$ and learn the value of x_s generated.

B Proof of Lemma 1

Before giving the proof of Lemma 1, we need the following pre-requisite notions. A process is called atomic if it can derived from the grammar:

$$A ::= 0^l \big| \nu x^l \big| (x := t)^l \big| [c_1 \wedge ... \wedge c_k]^l \big| in(x)^l \big| out(t)^l$$

where $c_i \in \{\top, s = t\}$ for all $i \in \{1, ..., k\}$. A process is called linear if it can be derived form the grammar $L ::= A | (L \cdot^l A)$. We will use a to denoted atomic processes.

Definition 8. *Let P be a process and $\mathcal{A}$ be an adversary for P. For any $\rho \in \mathsf{Exec}(\llbracket P \rrbracket^{\mathcal{A}})$, define the linear process $L(\rho)$ inductively as follows. For the base case $L((P, \emptyset, \emptyset)) = \epsilon$. For the inductive case, let $\rho = \rho_0 \xrightarrow{(\S,[l])} z'_n$. If l is the label of an atomic action a, then $L(\rho) = L(\rho_0) \cdot a$. Otherwise, if l does not label an atomic action, $L(\rho) = L(\rho_0)$.*

Proposition 5. *Let P be a process, $\mathcal{A}$ be an adversary for P and $\rho \in \mathsf{Exec}(\llbracket P \rrbracket^{\mathcal{A}})$ such that $last(\rho) = (Q, \varphi, \sigma)$. There exists an adversary $\mathcal{A}'$ for $L(\rho)$ and execution $\rho' \in \mathsf{Exec}(\llbracket L(\rho) \rrbracket^{\mathcal{A}'})$ where $last(\rho') = (Q', \varphi', \sigma')$ is such that*

1. *$dom(\varphi) = dom(\varphi')$ and $dom(\sigma) = dom(\sigma')$.*
2. *For all $x, y \in dom(\sigma)$, $x\sigma =_E y\sigma$ iff $x\sigma' =_E y\sigma'$*
3. *If $\varphi \vdash_E x\sigma$ then $\varphi' \vdash_E x\sigma'$*

Proposition 6. *For any equational theory E and $q \in [0, 1]$ and label set L, if $R = \nu z_1^{l_1} \cdot \nu z_2^{l_2} \cdot (x_1 := z_1)^{l_3} \cdot (x_2 := z_2)^{l_4} \cdot Q \not\models_{E,q} \mathsf{Secret}(S)$ and $l_1, l_2, l_3, l_4 \in L$ then $R' = \nu z^{l_1} \cdot (x_1 := z)^{l_2} \cdot (x_2 := z)^{l_3} \cdot Q \not\models_{E,q} \mathsf{Secret}(S)$ where $S \subseteq vars(Q)$ and $z, z_1, z_2 \notin vars(Q)$.*

We now give the proof of Lemma 1.

Proof. (Lemma 1) Let $I = C[B_1, ..., B_n]$ and

$$Q'' := \nu k_1^{l_0} \cdot ... \nu k_n^{l_n} \cdot (x_1^b := k_1)^{l_{n+1}} \cdot ... \cdot (x_n^b := k_n)^{l_m} \cdot C[B_1, ..., B_n]^b_{\mathcal{L}_b}.$$

By definition, there exists and adversary $\mathcal{A}$ such that $\llbracket I \rrbracket^{\mathcal{A}} \not\models_{E,q} \mathsf{Secret}(x_s)$. We transform A into an adversary $\mathcal{A}'$ for Q'' inductively as follows. If $\rho \in \mathsf{Exec}(\llbracket Q'' \rrbracket^{\mathcal{A}'})$ where $last(\rho) = (Q_z, \varphi_z, \sigma_z)$ is such that $Q_z = a^{l_k} \cdot ... \cdot a^{l_m} \cdot C[B_1, ..., B_n]^b_{\mathcal{L}_b}$ then $\mathcal{A}'(tr(\rho)) = (\tau, [l_k])$. Otherwise if $\rho = (Q, \emptyset, \emptyset) \xrightarrow{(\tau,[l_0])}$

... $\xrightarrow{(\tau,[l_m])} \rho_0$ then $\mathcal{A}'(tr(\rho)) = \mathcal{A}(tr(\rho_0))$. In the latter a case, we write $proj(\rho) = \rho_0$.

Let $\mathcal{R} \subseteq \mathsf{Exec}([\![Q'']\!]^{\mathcal{A}'})$ be such that $\rho \in \mathcal{R}$ iff either $last(\rho) \models_E \mathsf{Secret}(S \cup S^b \cup x_s)$ or $\rho = \rho_0 \xrightarrow{\alpha} z$ is such that $last(\rho) \not\models_E \mathsf{Secret}(S \cup S^b \cup x_s)$ and $last(\rho_0) \models_E \mathsf{Secret}(S \cup S^b \cup x_s)$. We claim that $[\![Q'']\!]^{\mathcal{A}'}|_{\mathcal{R}} \not\models_{E,q} \mathsf{Secret}(S \cup S^b \cup x_s)$. To see this, it suffices to show that for any $\rho \in \mathsf{Exec}([\![Q'']\!]^{\mathcal{A}'}|_{\mathcal{R}})$ such that $last(\rho) \not\models \mathsf{Secret}(S \cup S^b \cup x_s)$, $proj(\rho) \in \mathsf{Exec}(I^{\mathcal{A}})$ is such that $proj(\rho) \not\models \mathsf{Secret}(S \cup S^b \cup x_s)$. Assume for a contradiction that $last(\rho) \models_E \mathsf{Secret}(S \cup S^b \cup x_s)$ but $last(proj(\rho)) \not\models_E \mathsf{Secret}(S \cup S^b \cup x_s)$. By Proposition 5, we have $L(\rho) \models_E \mathsf{Secret}(S \cup S^b \cup x_s)$ and $L(proj(\rho)) \not\models_E \mathsf{Secret}(S \cup S^b \cup x_s)$. By the definitions of Q'' and I, $L(\rho)$ and $L(proj(\rho))$ meet the conditions of Theorem 1 from [18]. That is, $L(\rho) \not\models_E \mathsf{Secret}(S \cup S^b \cup x_s)$, contradiction. Therefore, $Q'' \not\models_{E,q} \mathsf{Secret}(S \cup S^b \cup x_s)$. By Proposition 6, we get $Q \not\models_{E,q} \mathsf{Secret}(S \cup S^b \cup x_s)$ as desired. $\square$

C Proof of Lemma 2

The main idea of the proof is to transform the recipes used in inputs from the context (resp. basic processes) to contain only frame variables previously output by the context (resp. basic processes). The formal definition of this transformation is given below.

Definition 9. *Let $N_{E,\tilde{n}} : \mathcal{T}(\mathcal{F}_b \cup \mathcal{F}_c) \to \mathcal{N}$ be a function such that if $t_1 =_E t_2$ then $N_{E,\tilde{n}}(t_1) = N_{E,\tilde{n}}(t_2) = n$ where $n \notin \tilde{n}$. For $d \in \{b, c\}$, define the function $Pur^{\varphi,\tilde{n}}_{E,d} : \mathcal{T}(\mathcal{F}_b \cup \mathcal{F}_c, \mathcal{X}_w) \to \mathcal{T}(\mathcal{F}_d, \mathcal{X}^d_w)$ as follows.*

$$Pur^{\varphi,\tilde{n}}_{E,d}(r) = \begin{cases} w_l & \text{if } r = w_{i,[l]} \text{ and } l \in \mathcal{L}_d \\ n & \text{if } r = w_{i,[l]},\ l \notin \mathcal{L}_d \text{ and } N_{E,\tilde{n}}(w_{i,[l]}\varphi) = n \\ f(Pur^{\varphi,\tilde{n}}_{E,d}(r_1), ..., Pur^{\varphi,\tilde{n}}_{E,d}(r_n)) & \text{if } r = f(r_1, ..., r_n) \text{ and } f \in \mathcal{F}_d \\ n & \text{if } r = f(r_1, ..., r_n), f \notin \mathcal{F}_d \text{ and } N_{E,\tilde{n}}(r\varphi) = n \end{cases}$$

Before applying the function from Definition 9 to transform recipes into pure recipes, we must first remove all self canceling function symbols from the recipe using the function *col* given below. Since we are considering equational theories that do not necessarily terminate, *col* will map terms to a kind of normal form. Recall that $(\mathcal{F}_b, E_b)$ and $(\mathcal{F}_c, E_c)$ are disjoint equational theories and $d \in \{b, c\}$. We will write $\overline{d}$ to denote $\{b, c\} \setminus d$. A ground term $t \in \mathcal{T}(\mathcal{F}_b \cup \mathcal{F}_c)$ is said to be *pure* if $t \in \mathcal{T}(\mathcal{F}_d)$. Similarly a context $r[y_1, ..., y_n] \in \mathcal{T}(\mathcal{F}_b \cup \mathcal{F}_c, \mathcal{X}_w \cup \mathcal{X})$, where $y_1, ..., y_n$ are the variables occurring in r, is called a pure $\mathcal{F}_d$-context if $r \in \mathcal{T}(\mathcal{F}_d, \mathcal{X}_w \cup \mathcal{X})$. Given a term t, we use *root*(t) to denote the signature of the root function symbol of t. Let t be a term such that $t = r[s_1, ..., s_m]$ for some a pure $\mathcal{F}_d$-recipe r and *root*$(s_1), ...,$ *root*$(s_m) \in \mathcal{F}_{\overline{d}}$. In such a case, we write $r[[s_1, ..., s_m]]$ and say that $s_1, ..., s_m$ are the *alien* subterms of t.

Definition 10. *Let $col(t) : \mathcal{T}(\mathcal{F}, \mathcal{X}_w) \rightarrow \mathcal{T}(\mathcal{F}, \mathcal{X}_w)$ be the function defined below.*

$$col(t) = \begin{cases} t & \text{if } t \text{ is a variable or a constant} \\ s_i & \text{if } t = f(t_1, ..., t_l), f(col(t_1), ..., col(t_l)) = r[[s_1, ..., s_k]], \\ & r[n_1, ..., n_k] =_{E_d} n_i \text{ where } n_i, ..., n_k \text{ are fresh names} \\ & \text{such that } n_i = n_j \text{ iff } s_i =_E s_j \text{ for all } 1 \leq i, j \leq l \\ & \text{and } r \text{ is a pure } \mathcal{F}_d\text{-recipe.)} \\ f(col(t_1), ..., col(t_l))) & \text{if } t = f(t_1, ..., t_l) \text{ but the above condition does not hold} \end{cases}$$

Lemma 3 ([18]). *(Fundamental Collapse Lemma) If $s =_E t$, then $col(s) = r_a[[s_1, ..., s_k]]$ and $col(t) = r_b[[s_{k+1}, ..., s_{k+l}]]$ are such that r_a and r_b are pure $\mathcal{F}_d$-terms, for $d \in \{b, c\}$, and $r_a[n_1, ..., n_k] =_{E_d} r_b[n_{k+1}, ..., n_{k+l}]$ where $n_1, ..., n_{k+l}$ are fresh names such that $n_i = n_j$ iff $s_i =_E s_j$ for all $1 \leq i, j \leq k + l$.*

Lemma 4 ([18]). *For any term t we have that $col(t) =_E t$.*

Lemma 5. *If $E = E_b \cup E_c$ and $r \in \mathcal{T}(\mathcal{F}_b \cup \mathcal{F}_c, dom(\varphi))$ is a recipe, then $col(r)\varphi =_E r\varphi$.*

Proof. Let $fv(r) = \{x_1, ..., x_n\}$ and $\varphi_c = \{x_1 \mapsto c_1, ..., x_n \mapsto c_n\}$ where $c_1, ..., c_n$ are constants. By Lemma 4, $r\varphi_c =_E col(r\varphi_c)$. By the definition of φ_c, $col(r\varphi_c) = col(r)\varphi_c$ and we have $r\varphi_c =_E col(r)\varphi_c$. It follows that $r\varphi = col(r)\varphi$. □

We now give the proof of Proposition 2.

Proof. (Proposition 2) Let $R = B_0 \cdot C[B_1, ..., B_n]$ and let $\tilde{n}$ be the set of names occurring in R. We define the adversary $\mathcal{A}'$ as follows. For any execution

$$\rho = (R, \emptyset, \emptyset) \xrightarrow{(r_1,[l_1]),...,(r_n,[l_n])}_* (R_n, \varphi_k, \sigma_k)$$

of R w.r.t $\mathcal{A}$ let

$$\rho' = (R, \emptyset, \emptyset) \xrightarrow{(r'_1,[l_1]),...,(r'_n,[l_n])}_* (R_n, \varphi'_k, \sigma'_k)$$

be the execution of R w.r.t. $\mathcal{A}'$ where $r'_i = Pur^{\varphi_k,\tilde{n}}_{E,b}(col(r_i))$ if recipe r_i comes from an action $(r_i, [l_i])$ such that $l_i \in L_b$ and $r'_i = Pur^{\varphi_k,\tilde{n}}_{E,c}(col(r_i))$ otherwise. It suffices to show that the following properties hold.

1. If r_i for $i \in \{1, ..., n\}$ is a recipe for $in(x)$ where $x \in \mathcal{X}_d$ and the frame is φ_j for $j \in \{1, ..., k\}$ then r'_i is a recipe from $\mathcal{T}(\mathcal{F}_d, \mathcal{X}^d_w)$. (Pure recipes)
2. If $x\sigma_j =_E y\sigma_j$ for $x, y \in \mathcal{X}_d$ then $x\sigma'_j =_{E_d} y\sigma'_j$. (Tests work)
3. If there exists a recipe r such that $r\varphi_j =_E x\sigma_j$ for $x \in \mathcal{X}_d$ then there exists a recipe r' such that $r'\varphi'_j =_{E_d} x\sigma'_j$. (Revealing the same secrets)
4. For any $\rho_1, \rho_2 \in \mathsf{Exec}(R^{\mathcal{A}'})$, if $tr(\rho_1) = tr(\rho_2)$ then $\mathcal{A}'(tr(\rho_1)) = \mathcal{A}'(tr(\rho_2))$. (Valid adversary)

□

Claim. If $x \in \mathcal{X}_d$ and $col(x\sigma_j) = r_x[[s_1, ..., s_n]]$ then $x\sigma'_j = r_x[N_{E,\tilde{n}}(s_1), ..., N_{E,\tilde{n}}(s_n)]$ where r_x is a pure $\mathcal{F}_d$-recipe.

Proof. The proof is by induction on j. The base case, when $j = 0$, follows trivially as $\sigma_0 = \sigma'_0 = \emptyset$. For the induction step, if $\sigma_j = \sigma_{j-1}$ the the result follows by the induction hypothesis. Otherwise the last action in the execution was an assignment or an input for some variable z. In the case of input, $\sigma_j = \sigma_{j-1} \cup \{z \mapsto r_z[[s_1, ..., s_n]]\}$ and $\sigma'_j = \sigma'_{j-1} \cup \{z \mapsto r_z[N_{E,\tilde{n}}(s_1), ..., N_{E,\tilde{n}}(s_n)]\}$ and the result follows by the induction hypothesis. Otherwise, if the last action was an assignment, then $\sigma_j = \sigma_{j-1} \cup \{z \mapsto r[[s_1, ..., s_n]]\}$ where $s_i = r_i\sigma_{j-1}$ for all $i \in \{1, ..., n\}$ and again the result follows from the induction hypothesis. □

Claim. Property 2 holds.

Proof. Given $x, y \in \mathcal{X}_d$ such that $x\sigma_j =_E y\sigma_j$, we want to show $x\sigma'_j =_{E_d} y\sigma'_j$. Let $col(x\sigma_j) = r_x[[s_1, ..., s_n]]$ and $col(y\sigma_j) = r_y[[s_{n+1}, ..., s_{n+l}]]$, where r_x, r_y are pure $\mathcal{F}_d$-recipes and $s_1, ..., s_{n+l}$ are alien subterms. By Lemma 4, $x\sigma_j =_E r_x[[s_1, ..., s_n]]$ and $y\sigma_j =_E r_y[[s_{n+1}, ..., s_{n+l}]]$. By the preceding claim, $x\sigma'_j = r_x[N_{E,\tilde{n}}(s_1), ..., N_{E,\tilde{n}}(s_n)]$ and $y\sigma'_j = r_y[N_{E,\tilde{n}}(s_{n+1}), ..., N_{E,\tilde{n}}(s_{n+1})]$. Applying Lemma 3, we have $r_x[N_{E,\tilde{n}}(s_1), ..., N_{E,\tilde{n}}(s_n)] =_{E_d} r_y[N_{E,\tilde{n}}(s_{n+1}), ..., N_{E,\tilde{n}}(s_{n+1})]$. That is $x\sigma'_j =_{E_d} y\sigma'_j$. □

Claim. Property 3 holds

Proof. Given $(R_j, \varphi_j, \sigma_j)$ and $(R'_j, \varphi'_j, \sigma'_j)$ we want to show that if $r\varphi_j =_E x\sigma_j$ for some $x \in dom(\sigma) \cap \mathcal{X}_d$ then $r'\varphi'_j =_{E_d} x\sigma'_j$ where $r' = Pur^{\varphi_k,\tilde{n}}_{E,d}(col(r))$. By Lemma 5, we have $col(r)\varphi_j =_E r\varphi_j$ where $col(r)\varphi_j = r_c[[s_1, ..., s_m]]$ and r_c is a pure $\mathcal{F}_d$-context. By Lemma 4, $x\sigma_j = col(x\sigma_j)$ where $col(\sigma_j = r_x[[s_{l+1}, ..., s_{l+m}]]$ and r_x is a pure $\mathcal{F}_d$-context. Applying Lemma 3, we get that $r_c[N_{E,\tilde{n}}(s_1), ..., N_{E,\tilde{n}}(s_m)] =_{E_d} r_x[N_{E,\tilde{n}}(s_{m+1}), ..., N_{E,\tilde{n}}(s_{m+l})]$. By the definition of r', we have $r'\varphi'_j = r_c[N_{E,\tilde{n}}(s_1), ..., N_{E,\tilde{n}}(s_m)]$. Further, by our earlier claim, $x\sigma'_j = r_x[N_{E,\tilde{n}}(s_{m+1}), ..., N_{E,\tilde{n}}(s_{m+l})]$. That is, $r'\varphi'_j =_{E_d} x\sigma'_j$. □

Claim. Property 4 holds

Proof. Assume for a contradiction that there exists $\rho_1, \rho_2 \in \mathsf{Exec}(R^{\mathcal{A}'})$ such that $tr(\rho_1) = tr(\rho_2)$ and $\mathcal{A}'(tr(\rho_1)) \neq \mathcal{A}'(tr(\rho_2))$. Because $\mathcal{A}$ is an adversary, we have $\mathcal{A}(tr(\rho_1)) = (\S, [l]) = \mathcal{A}(tr(\rho_2))$. If $\S = \tau$, then $\mathcal{A}'(tr(\rho_1)) = (\tau, [l]) = \mathcal{A}'(tr(\rho_2))$, contradiction. If $\S = r$, for some recipe r, then $\mathcal{A}'(tr(\rho_1)) = (r_1, [l])$ and $\mathcal{A}'(tr(\rho_1)) = (r_2, [l])$. Now $l \in \mathcal{L}_d$ for $d \in \{b, c\}$ and hence $r_1 = Pur^{\varphi_k,\tilde{n}}_{E,d}(col(r))$ and $r_2 = Pur^{\varphi'_k,\tilde{n}}_{E,d}(col(r))$ for frames $\varphi^k, \varphi^{k'}$. Because $\varphi^k \approx \varphi^{k'}$ we have $r_1 = r_2$, contradiction. □

D Proof of Theorem 1

We first fix some notation that will be used throughout the remainder of this section. Let $M_i = (Z_i, z^s_i, Act_i, \Delta_i, \equiv_i)$ be a POMDP for $i \in \{1, 2\}$. We will assume that $Z_1 \cap Z_2 = \emptyset$ and $Act_1 \cap Act_2 = \emptyset$. The asynchronous product of M_1 and M_2, denoted $M_1 \otimes M_2$ is the POMDP $(Z, z_s, Act, \Delta, \equiv)$ where $Z = \{(z_1, z_2) \mid z_1 \in Z_1 \wedge z_2 \in Z_2\}$, $z_s = (z_1^s, z_2^s)$, $Act = Act_1 \cup Act_2$, $\Delta((z_1, z_2), \alpha_1) = \Delta_i(z_i, \alpha_i)$ and $(z_1, z_2) \equiv (z_1', z_2')$ iff $z_1 \equiv_1 z_1'$ and $z_2 \equiv_2 z_2'$. Let $\mathcal{A}$ be an adversary for $M_1 \otimes M_2$. We will assume that $\mathsf{Exec}((M_1 \otimes M_2)^{\mathcal{A}})$ is finite and every execution is of finite length. For $\rho \in \mathsf{Exec}((M_1 \otimes M_2)^{\mathcal{A}})$, we define its projection onto M_i, denoted $proj(\rho, M_i)$ inductively as follows. When $\rho = (z_1^s, z_2^s)$, $proj(\rho, M_i) = z_i^s$. When $\rho = \rho_0 \xrightarrow{\alpha} (z_1, z_2)$, $proj(\rho, M_i) = proj(\rho_0, M_i) \xrightarrow{\alpha} z_i$ if $\alpha \in Act(M_i)$ and otherwise $proj(\rho, M_i) = proj(\rho_0, M_i)$ if $\alpha \notin Act(M_i)$. Let $S_1 \subseteq Z_1$, $S_2 \subseteq Z_2$ and $S = S_1 \cup S_2$. We write $prob(S, (M_1 \otimes M_2)^{\mathcal{A}})$ to denote the maximal p such that $(M_1 \otimes M_2)^{\mathcal{A}} \models_p S$. Intuitively, S can be thought of the attack states (states in which the attacker can derive some secret value) in $M_1 \otimes M_2$ and $prob(S, (M_1 \otimes M_2)^{\mathcal{A}})$ is the exact probability of reaching an attack state. An execution $\rho \in (M_1 \otimes M_2)^{\mathcal{A}}$ is said to be distinguishing if there exist one step extensions ρ_1 and ρ_2 of ρ such that $last(\rho_1) \not\equiv last(\rho_2)$. An $l \in \mathbb{N}$ is said to be a distinguishing level in $(M_1 \otimes M_2)^{\mathcal{A}}$ if $l = min(|\rho|)$ for all $\rho \in \mathsf{Exec}((M_1 \otimes M_2)^{\mathcal{A}})$. When $(M_1 \otimes M_2)^{\mathcal{A}}$ contains no distinguishing executions, the distinguishing level is ∞. For ease of notation, we will write $\bar{i}$ to denote the only element of $\{1, 2\} \setminus \{i\}$.

Definition 11. *An attacker $\mathcal{A}$ for $M_1 \otimes M_2$ is said to be process determined if, for any $\rho, \rho' \in \mathsf{Exec}((M_1 \otimes M_2)^{\mathcal{A}})$, if $tr(proj(\rho, M_i)) = tr(proj(\rho', M_i))$ and $\mathcal{A}(tr(\rho)) \in Act(M_i)$ then $\mathcal{A}(tr(\rho)) = \mathcal{A}(tr(\rho'))$.*

Definition 12. *An execution $\rho \in \mathsf{Exec}((M_1 \otimes M_2)^{\mathcal{A}})$ is said to be $(M_i, M_{\bar{i}})$-sequential if there exists a k such that*

$$\rho = z_0 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_k} z_k \xrightarrow{\alpha_{k+1}} \ldots \xrightarrow{\alpha_{k+l}} z_{k+m}$$

where $\alpha_1, \ldots, \alpha_k \in Act_i$ and $\alpha_{k+1}, \ldots, \alpha_{k+m} \in Act_{\bar{i}}$. The attacker $\mathcal{A}$ is called $(M_i, M_{\bar{i}})$-sequential if any execution in $\mathsf{Exec}((M_i \otimes M)^{\mathcal{A}})$ is $(M_i, M_{\bar{i}})$-sequential.

Lemma 6. *For any $(M_i, M_{\bar{i}})$-sequential attacker $\mathcal{A}$ of $M_1 \otimes M_2$, there exists a sequential and process determined attacker $\mathcal{A}'$ such that $prob(S, (M_1 \otimes M_2)^{\mathcal{A}}) \leq prob(S, (M_1 \otimes M_2)^{\mathcal{A}'})$.*

Proof. Let $\rho = z_0 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_k} z_k$ be an execution in $\mathsf{Exec}((M_1 \otimes M_2)^{\mathcal{A}})$ such that $\alpha_1, \ldots, \alpha_k \in Act_1$ and for any one step extension $\rho' = \rho \xrightarrow{\alpha} z$, $\alpha \in Act_2$. We know that for any pair of one step extensions ρ_1 and ρ_2 of ρ, $tr(proj(\rho_1, M_2)) = tr(proj(\rho_2, M_2))$. That is, we can define an attacker $\mathcal{A}'$ on M_2 with respect to the execution ρ by induction on the number of transitions in the MDP $(M_1 \otimes M_2)^{\mathcal{A}}$ starting from the prefix ρ. For the base case, let $z_2 \in \mathsf{Exec}((proj(\rho, M_2))^{\mathcal{A}'})$ where $last(\rho) = (z_1, z_2)$. For the inductive step, let $\rho \xrightarrow{\alpha} \rho' \xrightarrow{\alpha'} z$ be an execution of $(M_1 \otimes M_2)^{\mathcal{A}}$. Inductively, we have an execution $\rho'' \in \mathsf{Exec}((M_1 \otimes M_2)^{\mathcal{A}'})$ for $\rho \xrightarrow{\alpha} \rho'$. Define $\mathcal{A}'(tr(\rho'')) = \alpha'$.

Let Θ be the set of all executions in $\mathsf{Exec}((M_1 \otimes M_2)^{\mathcal{A}})$ having the property of execution ρ. For each $\rho_j \in \Theta$, let $\mathcal{A}_j$ be the scheduler for M_2 defined with respect to ρ_i. Let $j \in \{1, ..., n\}$ and m be the index of some execution in Θ such that $prob(S, M_2^{\mathcal{A}_m}) \geq prob(S, M_2^{\mathcal{A}_j})$ for all j. Define the attacker $\mathcal{A}''$ for $M_1 \otimes M_2$ that behaves like $\mathcal{A}$ until reaching an execution of Θ and then behaves like $\mathcal{A}_m$ on the remaining M_2 component. Clearly, $\mathcal{A}''$ is both sequential and process determined. Furthermore, because M_2 is executed with respect to the maximal adversary $\mathcal{A}_m$ of M_2 for every execution in $(M_1 \otimes M_2)^{\mathcal{A}''}$, we have $prob(S, (M_1 \otimes M_2)^{\mathcal{A}}) \leq prob(S, (M_1 \otimes M_2)^{\mathcal{A}''})$. □

Proposition 7. *Let l be the distinguishing level of $(M_1 \otimes M_2)^{\mathcal{A}}$ and $k \leq l$. There exists an attacker $\mathcal{A}'$ for $M_1 \otimes M_2$ such that for any $\rho \in \mathsf{Exec}(M_1 \otimes M_2)$ where $|\rho| = k$, ρ is $(M_i, M_{\bar{i}})$-sequential and $prob(S, (M_1 \otimes M_2)^{\mathcal{A}}) = prob(S, (M_1 \otimes M_2)^{\mathcal{A}'})$.*

Lemma 7. *For any attacker $\mathcal{A}$ of $M_1 \otimes M_2$, there exists an (M_1, M_2)-sequential and processes determined attacker $\mathcal{A}'$ such that $prob(S, (M_1 \times M_2)^{\mathcal{A}}) \leq prob(S, (M_1 \otimes M_2)^{\mathcal{A}'})$.*

Proof. The proof is by induction on the number of distinguishing executions in the MDP $(M_1 \otimes M_2)^{\mathcal{A}}$. The base case, when there are no distinguishing executions, follows by Proposition 7 and Lemma 6. For the inductive step, let the number of distinguishing executions in $(M_1 \otimes M_2)^{\mathcal{A}}$ be $d + 1$ and let the distinguishing level be l. For any $\rho_1, \rho_2 \in (M_1 \otimes M_2)^{\mathcal{A}}$ where $|\rho_1| = |\rho_2| = l$, $\mathcal{A}(tr(\rho_1)) = \mathcal{A}(tr(\rho_2))$. Fix $\mathcal{A}(tr(\rho_1)) = \alpha$. We proceed by cases.

Case 1: $\alpha \in Act_1$. By Proposition 7, there exists an adversary $\mathcal{A}'$ such that for any $\rho \in \mathsf{Exec}((M_1 \otimes M_2)^{\mathcal{A}'})$ where $|\rho| = l$, ρ is (M_1, M_2)-sequential and $prob(S, (M_1 \otimes M_2)^{\mathcal{A}}) = prob(S, (M_1 \otimes M_2)^{\mathcal{A}'})$. Let the distinguishing level of this new DTMC $(M_1 \otimes M_2)^{\mathcal{A}'}$ be l' and let $\Theta = \{\rho \,|\, \rho \in \mathsf{Exec}((M_1 \otimes M_2)^{\mathcal{A}'})$ and $|\rho| = l'\}$. Assume without loss of generality that $\Theta = \Theta_{e_1} \uplus \Theta_{e_1}$ where $\Theta_{e_i} = \{\rho \mid \rho \in \Theta$ and $last(\rho) \in e_i\}$ for some e_i in $\equiv$. Let $f_i : \{1, ..., |\Theta_{e_i}|\} \mapsto \Theta_{e_1}$ be a bijection. For $j \in \{1, ..., |\Theta_{e_i}|\}$, let κ_j be the probability of event $f_i(j)$ in $(M_1 \otimes M_2)^{\mathcal{A}'}$ and let κ_{e_i} be the sum of the probabilities of the events $f_i(1), ..., f_i(j)$. Define the POMDP $M_{e_i} = (Z_i, z_i^s, Act_i \cup \{\alpha_{new}\}, \Delta_i', \equiv_i)$, where $\alpha_{new} \notin Act_1 \cup Act_2$ and Δ_i' is the same as Δ_i with addition of $\Delta_i'(z_i^s, \alpha_{new}) = \mu$ where $\mu(last(f_i(j))) = \kappa_j / \kappa_{e_i}$ for all j. We define an attacker $\mathcal{A}_{e_i}$ on $M_{e_i} \otimes M_2$ from $\mathcal{A}'$ such that

$$z_s \xrightarrow{\alpha_{new}} z_{l'} \xrightarrow{\alpha_{l'+l}} ... \xrightarrow{\alpha_{l'+m}} z_{l'+m} \tag{1}$$

is an execution of $(M_{e_i} \otimes M_2)^{\mathcal{A}_{e_i}}$ iff

$$z_0 \xrightarrow{\alpha_1} ... \xrightarrow{\alpha_{l'}} z_{l'} \xrightarrow{\alpha_{l'+l}} ... \xrightarrow{\alpha_{l'+m}} z_{l'+m} \tag{2}$$

is an execution of $(M_i \otimes M_2)^{\mathcal{A}_i'}$. Notice that the number of distinguishing states in $(M_{e_i} \otimes M_2)^{\mathcal{A}_{e_i}}$ is d. By our inductive assumption, there exists a process determined and (M_1, M_2)-sequential attacker $\mathcal{A}_{e_i}'$ for $M_{e_i} \otimes M_2$ where $prob(S, (M_{e_i} \otimes M_2)^{\mathcal{A}_{e_i}}) \leq prob(S, (M_{e_i} \otimes M_2)^{\mathcal{A}'_{e_i}})$. Using $\mathcal{A}_{e_1}'$ and $\mathcal{A}_{e_2}'$, we construct a scheduler $\mathcal{A}_e$ for $M_1 \otimes M_2$ inductively as follows. For any $\rho \in (M_1 \otimes M_2)^{\mathcal{A}_e}$ such that $|\rho| < l'$, let $\mathcal{A}_e(tr(\rho)) = \mathcal{A}'(tr(\rho))$. For any $\rho \in (M_1 \otimes M_2)^{\mathcal{A}_e}$ such that $|\rho| \geq l'$, $\mathcal{A}_e$ behaves like $\mathcal{A}'_{e_i}$ if the initial prefix of ρ is in Θ_{e_i}. Clearly, $prob(S, (M_1 \otimes M_2)^{\mathcal{A}'}) \leq prob(S, (M_1 \otimes M_2)^{\mathcal{A}_e})$. Observe that $\mathcal{A}_e$ is a sequential adversary for $M_1 \otimes M_2$. Because $prob(S, (M_1 \otimes M_2)^{\mathcal{A}}) = prob(S, (M'_1, M_2)^{\mathcal{A}'}) \leq prob(S, (M_1 \otimes M_2)^{\mathcal{A}_e})$ we can apply Lemma 6 to conclude that there exits a sequential and process determined adversary $\mathcal{A}'_e$ for $M_1 \otimes M_2$ such that $prob(S, (M_1 \otimes M_2)^{\mathcal{A}}) \leq prob(S, (M_1, M_2)^{\mathcal{A}'_e})$.

Case 2: $\alpha \in Act(M_2)$. Follows by a similar argument as case 1. $\square$

Proposition 8. *If $M_1 \models_{q_1} S_1$ and $M_2 \models_{q_2} S_2$ then for any (M_1, M_2)-sequential and process determined scheduler $\mathcal{A}$ for $M_1 \otimes M_2$, $(M_1 \otimes M_2)^{\mathcal{A}} \models_{q_1 q_2} S$.*

Proposition 9. *Let $C[\square_1, ..., \square_n]$ be a context over $\mathcal{F}_c$ with labels from $\mathcal{L}_c$ and $B_1, ..., B_n$ be basic processes over $\mathcal{F}_b$ with labels from $\mathcal{L}_b$. For $S \subseteq vars(C[B_1, ..., B_n])$, let $\mathcal{A}$ be an adversary for $C[B_1, ..., B_n]$ that is pure with respect to $(\mathcal{L}_b, \mathcal{F}_b)$ and $(\mathcal{L}_c, \mathcal{F}_c)$ such that $[\![C[B_1, ..., B_n]]\!]^{\mathcal{A}} \not\models_q \mathsf{Secret}(S)$. For $B = B_1|...|B_n$, there exists an adversary $\mathcal{A}'$ for $[\![C[[\top]^{l_1}, ..., [\top]^{l_n}]]\!] \otimes [\![B]\!]$, where $l_1, ..., l_n$ are fresh labels from $\mathcal{L}_b$, such that $([\![C[[\top]^{l_1}, ..., [\top]^{l_n}]]\!] \otimes [\![B]\!])^{\mathcal{A}'} \not\models_q \mathsf{Secret}(S)$.*

We now give the proof of Theorem 1.

Proof. (Theorem 1) Let $S = \{x_1, ..., x_n\}$ and $Q = \nu k \cdot (x_1 := k) \cdot ... \cdot (x_n := k) \cdot (B_1|...|B_n)$. Assume for a contradiction that there exits an adversary $\mathcal{A}$ such that $[\![C[B_1, ..., B_n]]\!]^{\mathcal{A}} \not\models_{E,q_1q_2} \mathsf{Secret}(x_s)$. By Lemma 1, their exists an adversary $\mathcal{A}'$ for the process

$$R := \nu k \cdot (x_1^b := k) \cdot ... \cdot (x_n^b := k) \cdot (C[B_1, ..., B_n])^b_{\mathcal{L}_b}$$

such that $[\![R]\!]^{\mathcal{A}'} \not\models_{E,q_1q_2} \mathsf{Secret}(S \cup S^b \cup x_s^b)$. By Proposition 2, the adversary $\mathcal{A}'$ can be transformed into an adversary $\mathcal{A}_p$ for R such that $[\![R]\!]^{\mathcal{A}_p} \not\models_{E,q_1q_2} \mathsf{Secret}(S \cup S^b \cup x_s^b)$ where $\mathcal{A}_p$ is pure with respect to $(\mathcal{L}_b, \mathcal{F}_b)$ and $(\mathcal{L}_c, \mathcal{F}_c)$. By Proposition 9, there exists an adversary $\mathcal{A}'_p$ for the POMDPs $M_1 = (C[[\top]^{l_1}, ..., [\top]^{l_n}])$ and $M_2 = \nu k \cdot (x_1^b := k) \cdot ... \cdot (x_n^b := k) \cdot ((B_1)^b_{\mathcal{L}_b}|...|(B_n)^b_{\mathcal{L}_b})$ such that $(M_1 \otimes M_2)^{\mathcal{A}'_p} \not\models_{q_1,q_2} \mathsf{Secret}(S)$. The result follows by Proposition 8. $\square$