Composition Theorems for CryptoVerif and Application to TLS 1.3

Bruno Blanchet

Composition Theorems for CryptoVerif and Application to TLS 1.3

Bruno Blanchet

ISSN 0249-6399 ISRN INRIA/RR-9171--FR+ENG

RESEARCH REPORT
N° 9171
April 2018
Project-Team Prosecco

Composition Theorems for CryptoVerif and Application to TLS 1.3

Bruno Blanchet

Project-Team Prosecco

Research Report n° 9171 — April 2018 — 67 pages

Abstract: We present composition theorems for security protocols, to compose a key exchange protocol and a symmetric-key protocol that uses the exchanged key. Our results rely on the computational model of cryptography and are stated in the framework of the tool CryptoVerif. They support key exchange protocols that guarantee injective or non-injective authentication. They also allow random oracles shared between the composed protocols. To our knowledge, they are the first composition theorems for key exchange stated for a computational protocol verification tool, and also the first to allow such flexibility.
As a case study, we apply our composition theorems to a proof of TLS 1.3 Draft-18. This work fills a gap in a previous paper that informally claims a compositional proof of TLS 1.3, without formally justifying it.

Key-words: composition, security protocols, verification, computational model, TLS

RESEARCH CENTRE
PARIS

2 rue Simone Iff
CS 42112
75589 Paris Cedex 12

Théorèmes de composition pour CryptoVerif et application à TLS 1.3

Résumé : Nous présentons des théorèmes de composition pour les protocoles cryptographiques, pour composer un protocole d'échange de clés et un protocole à clé symétrique qui utilise la clé échangée. Nous résultats reposent sur le modèle calculatoire de la cryptographie et sont formulés dans le cadre de l'outil CryptoVerif. Ils autorisent des protocoles d'échange de clés qui garantissent l'authentification injective ou non-injective. Ils autorisent aussi le partage d'oracles aléatoires entre les protocole composés. À notre connaissance, ils sont les premiers théorèmes de composition pour l'échange de clés formulés pour un outil de vérification de protocole dans le modèle calculatoire, et aussi les premiers à autoriser une telle flexibililté.

Comme étude de cas, nous appliquons nos théorèmes de composition à une preuve de TLS 1.3 brouillon 18. Ce travail fournit un élément manquant dans un article précédent qui donne informellement une preuve compositionnelle de TLS 1.3, sans la justifier formellement.

Mots-clés : composition, protocoles cryptographiques, vérification, modèle calculatoire, TLS

1 Introduction

The proof of security protocols is notoriously difficult. In particular, when security protocols grow in size, the complexity of their proof increases, and quickly becomes unmanageable. Composition theorems are essential to tackle this problem: they allow one to prove small pieces of the considered protocol, and to compose these results in order to obtain a proof of the full protocol.

In this paper, we consider the standard game-based computational model of cryptography, and we focus on the composition between a key exchange protocol and a symmetric key protocol that uses the key provided by the key exchange. We assume that the key exchange protocol runs between two participants A and B and is secure: the provided key is secret in the sense that keys provided in several sessions are indistinguishable from independent random keys; the protocol provides authentication of A to B, defined based on session identifiers; and A executes at most one session with a given session identifier. Then, we prove that the security properties of the symmetric key protocol carry over to the composed protocol. Moreover, we also prove that the security properties of the key exchange protocol are preserved in the composed protocol (except for the secrecy of the key on which we perform the composition). This point is important to be able to perform several compositions one after the other. We have two variants of our composition theorem: one in which authentication is injective, that is, each execution of B corresponds to a distinct execution of A; and one in which authentication is non-injective, that is, several executions of B may correspond to the same execution of A.

An originality of our composition theorems is that they are stated within the framework of a protocol verification tool, CryptoVerif [11, 12, 14], available at `http://cryptoverif.inria.fr`. We use the language of CryptoVerif to represent the cryptographic games; we use security properties proved by CryptoVerif as assumptions of our composition theorems. Therefore, we can easily use CryptoVerif to mechanize the proof of the protocol pieces that we compose.

Using such a framework to state composition theorems has several advantages. It strengthens the abilities of CryptoVerif: thanks to the composition theorems, we can obtain security proofs for bigger protocols. Moreover, CryptoVerif does not support loops. If we can break a protocol with loops into pieces without loops, we can verify them using CryptoVerif, and prove security of the whole protocol with loops by iteratively composing these pieces. Finally, CryptoVerif provides a rigorous framework for stating the composition theorems: it provides a formal language for games and for security properties, with a formal semantics. The hypotheses and conclusions of the composition theorems can then be stated precisely and concisely in that framework.

As a case study, we apply our composition theorems to TLS 1.3. More precisely, we revisit a previous analysis of TLS 1.3 Draft 18 by Bhargavan et al. [9,10]. This analysis splits TLS 1.3 into 3 pieces: the initial handshake, the handshake with pre-shared key, and the record protocol, and claims that security of TLS 1.3 can be obtained by composing these 3 pieces. However, it does not justify this composition formally. Our work fills this gap. TLS 1.3 is particularly well-suited as a case study to illustrate the power of our composition results. First, it is an important protocol, which is currently being standardized. The current draft, Draft 28 [35], is now final, and not very different from Draft 18. We expect that our composition results would apply in the same way to Draft 28, though the CryptoVerif models of the protocol pieces would require minor changes. Second, TLS 1.3 is well-designed to allow composition: the handshake produces traffic secrets used by the record protocol as well as a resumption secret used as a pre-shared key by the next handshake. The protocol pieces are cleanly separated, so that the only common secret between them is the symmetric key on which we perform the composition. Third, TLS 1.3 includes loops: it allows an unbounded number of handshakes with pre-shared key and an unbounded number of key updates in the record protocol. Therefore, it cannot be analyzed as a whole by CryptoVerif. The composition results allow us to break these loops and obtain security results for the full protocol. Finally, TLS 1.3 includes a variety of compositions, and we provide theorems for all of them. While most compositions use a key exchange that provides injective authentication, TLS also includes 0-RTT (Round Trip Time) data, that is, data that the client sends to the server immediately after the first message of TLS (`ClientHello`). Such data can be replayed, and the corresponding key exchange only provides non-injective authentication. Furthermore, we also have to deal with altered `ClientHello` messages; in this case, only the server has the corresponding key, and that requires a variant of the composition result. Finally, for key updates in the record protocol, the key is simply computed from the previous key without a proper key exchange, so we can use a much simpler composition theorem in this case.

Related Work The line of work closest to ours is that of [16, 27, 29]. Brzuska et al. [16] prove a composition result similar to ours, in an informal game-based framework. Fischlin et al. [27, 29] extend this framework to multi-stage key exchange protocols, in which parties can establish multiple keys in different stages and use these keys between stages. They apply their results to the proof of QUIC [29] and of TLS 1.3 Draft 05, Draft DH [27], and Draft 10 [28]. In addition to recasting the composition result in CryptoVerif, we extend it in several ways. We prove that security properties of the key exchange protocol are preserved in the composition, so we can compose again using other keys. This point appears in [29, Remark, page 16 of the full version] without proof. We allow the composed protocols to share random oracles; this point does not appear in [16, 27, 29]. We prove a composition theorem for protocols that provide non-injective authentication, used for 0-RTT data in TLS 1.3. Although we do not consider several stages explicitly, our composition theorems support most compositions allowed by the multi-stage framework. (We detail the comparison in Section 5.4.)

Brzuska et al. [15] prove a composition theorem that allows the key exchange protocol to already use the key provided to the symmetric-key protocol, for example for key confirmation. TLS 1.3 is designed so that the same key is never both used in the key exchange and provided to the next protocol, so we did not need such a composition result in our case study. We believe that such a result could also be proved in our framework if desired.

Canetti and Krawczyk [19] prove security of the composition of a key exchange protocol with specific symmetric-key protocols that use MACs to achieve an authenticated channel or encrypt-then-MAC to achieve a secure channel.

Barthe et al. [4] develop generic proofs of reduction for one-round key-exchange protocols, such as Naxos and HMQV, in EasyCrypt. EasyCrypt is an interactive theorem prover specialized for building game-based security proofs. Thus, their approach provides another way of introducing modularity in machine-checked proofs of security.

Universal composability (UC) [17,18,20,32] is a framework that allows to compose protocols. However, proving UC-security requires stronger properties than the game-based framework that we use ([16, Appendix A] details limitations of the UC framework). Delaune et al. [25] present a simulation-based framework that is an analogue of UC in the symbolic (Dolev-Yao) model of cryptography.

Composition theorems have also been proved in the Dolev-Yao model. Many of these theorems deal with the parallel composition of protocols that share secrets, for trace properties [22, 31], for resistance against guessing attacks for protocols that share passwords [26], and for privacy properties [2], using disjointness assumptions such as tagging or disjoint primitives to guarantee the independence of the protocols. Other results [3, 21] allow sequential composition, in particular the composition of a key exchange protocol with a symmetric-key protocol that uses the exchanged key. Ciobâcă et al. [21] consider trace properties, while Arapinis et al. [3] extend the result to processes with else branches, to private channels, and to privacy properties. Mödersheim et al. [30, 33, 34] define notions of security for channels (insecure, authentic, confidential, secure), and prove composition results between protocols that establish such channels and protocols that use them. They also rely on the Dolev-Yao model and use disjointness assumptions. We believe that the computational model has two advantages: it is more realistic than the Dolev-Yao model and the computational definitions compose nicely, so that we can avoid disjointness assumptions.

Protocol composition logic [23,24] is a logic for proving security protocols that allows sequential and parallel composition. It was initially designed in the Dolev-Yao model [23] and adapted to the computational model [24].

Outline The next section provides a minimal reminder of the CryptoVerif framework. Section 3 presents the structure of our proof of TLS, so that we can use it as a motivation for the composition theorems. Section 4 presents a very simple composition theorem, used for key updates in TLS 1.3, as a warm-up. Section 5 presents our main composition theorems. Section 6 summarizes their application to TLS, and Section 7 concludes. The appendix provides the proofs of all results and details on the TLS case study.

2 A Short Reminder on CryptoVerif

Processes, contexts, adversaries CryptoVerif mechanizes proofs by sequences of games, similar to those written on paper by cryptographers [8,36]. It represents protocols and cryptographic games in a probabilistic process calculus. We refer the reader to [12,14] for details on this process calculus. We explain the necessary constructs as they appear.

We use P, Q for *processes*. A *context* C is a process with one or several holes $[\,]$. We write $C[P_1, \dots, P_n]$ for the process obtained by replacing the holes of C with $P_1, \dots, P_n$ respectively. An *evaluation context* is a context with one hole, generated by the following grammar:

$C ::=$	evaluation context
$[\,]$	hole
newChannel $c; C$	channel restriction
$Q \mid C$	parallel composition
$C \mid Q$	parallel composition

The channel restriction **newChannel** $c; Q$ restricts the channel name c, so that communications on this channel can occur only inside Q, and cannot be received outside Q or sent from outside Q. The parallel composition $Q_1 \mid Q_2$ makes simultaneously available the processes defined in Q_1 and Q_2. We use evaluation contexts to represent *adversaries*.

Indistinguishability A process can execute events, by two constructs: **event** $e(M_1, \dots, M_n)$ executes event e with arguments $M_1, \dots, M_n$, and **event_abort** e executes event e without argument and aborts the game. After finishing execution of a process, the system produces two results: the sequence of executed events $\mathcal{E}$, and the information whether the game aborted ($a = $ **abort**, that is, executed **event_abort**) or terminated normally ($a = 0$). These events and result can be used to distinguish games, so we introduce an additional algorithm, a *distinguisher* D that takes as input the sequence of events $\mathcal{E}$ and the result a, and returns true or false. We write $\Pr[Q : D]$ for the probability that the process Q executes events $\mathcal{E}$ and returns a result a such that $D(\mathcal{E}, a) = \mathsf{true}$.

Definition 1 (Indistinguishability). *We write $Q \approx_p^V Q'$ when, for all evaluation contexts C acceptable for Q and Q' with public variables V and all distinguishers D that run in time at most t_D, $|\Pr[C[Q] : D] - \Pr[C[Q'] : D]| \le p(C, t_D)$.*

Intuitively, $Q \approx_p^V Q'$ means that an adversary has probability at most p of distinguishing Q from Q', when it can read the variables in the set V. When V is empty, we omit it. The probability p may depend on many parameters coming from the context C, that is why p takes as arguments the whole context C and the runtime of D. CryptoVerif always expresses the probabilities as formulas in which the only parameters that come from the context C are the maximum runtime of C, the maximum number of times C may send a message to each subprocess in Q (resp. Q'), and the lengths of bitstrings. This property allows us to simplify probability formulas by abstracting away the precise context they use and retaining only the useful parameters. We denote by t_C the maximum runtime of C, and use the same notation for processes P, Q, terms M, and functions f. As usual in cryptographic proofs, we ignore very small runtimes.

The condition that C is acceptable for Q and Q' with public variables V is a technical condition that ensures that $C[Q]$ and $C[Q']$ are well-formed. The public variables V are the variables of Q and Q' that C is allowed to read.

Indistinguishability is reflexive ($Q \approx_0^V Q$), symmetric (if $Q \approx_p^V Q'$, then $Q' \approx_p^V Q$) and transitive (if $Q \approx_p^V Q'$ and $Q' \approx_{p'}^V Q''$, then $Q \approx_{p+p'}^V Q''$). Moreover, $Q \approx_p^V Q'$ implies $C[Q] \approx_{p'}^{V'} C[Q']$ for all evaluation contexts C acceptable for Q and Q' with public variables V and all $V' \subseteq V \cup \mathsf{var}(C)$, where $\mathsf{var}(C)$ is the set of variables of C and $p'(C', t_D) = p(C'[C[\,]], t_D)$.

Secrecy Intuitively, in CryptoVerif, secrecy means that the adversary cannot distinguish between the secrets and independent random values. This definition corresponds to the "real-or-random" definition of security [1]. As shown in [1], this notion is stronger than the one in which the adversary performs a single test query and some reveal queries. We recall the definition of secrecy in CryptoVerif given in [13,14]. Let us first explain CryptoVerif constructs used in this definition. The *replication* $!^{i \leq n} Q$ represents n copies of the process Q in parallel, indexed by $i \in [1, n]$, where n is named a *replication bound*. The *current replication indices* at a certain program point are the indices i of replications above that program point. In CryptoVerif, all variables defined under a replication are implicitly arrays indexed by the current replication indices: if Q defines a variable x under $!^{i \leq n}$, the value of x is in fact stored in $x[i]$. The definition of x is executed at most once for each i, so that all values of x are stored in distinct array cells. When a variable is accessed with current replication indices, we omit the indices, writing x for $x[i]$. The **find** *construct* reads these array cells: **find** $u = i' \leq n$ **suchthat defined**$(x_1[i'], \dots, x_m[i']) \wedge M$ **then** P **else** P' looks for an index $i' \in [1, n]$ such that $x_1[i'], \dots, x_m[i']$ are defined and M is true. When such an index is found, it is stored in u, and process P is executed. Otherwise, process P' is executed. The term M may refer to $x_1[i'], \dots, x_m[i']$ and the process P may refer to $x_1[u], \dots, x_m[u]$ since these variables are guaranteed to be defined. The *input* $c[i](x : T); P$ receives a message on channel $c[i]$. If this message is in the set of bitstrings T (T stands for "type"), it is stored in x, and P is executed. Otherwise, the process blocks. The channel $c[i]$ consists of a channel name c and indices, here i. Very often, these indices are the current replication indices at the input: the sender can then tell precisely to which copy of the process the message should be sent. Similarly, the *output* $\overline{c[i]}\langle M \rangle; Q$ sends message M on channel $c[i]$. After the output, the control is passed to the receiver process, which continues execution. The process Q that follows the output consists of inputs, possibly under replications and parallel compositions; these inputs will be executed when a message is sent to them. Finally, the *restriction* **new** $y : T; P$ chooses uniformly a random element of T, stores it in y, and executes P.

We use $\widetilde{u}$ as an abbreviation for a sequence of variables: $\widetilde{u} = u_1, \dots, u_m$. We write $\widetilde{u} \leq \widetilde{n}$ for $u_1 : [1, n_1], \dots, u_m : [1, n_m]$ when $\widetilde{u} = u_1, \dots, u_m$ and $\widetilde{n} = n_1, \dots, n_m$. We say that a variable is defined under replications $!^{\widetilde{i} \leq \widetilde{n}}$ when $\widetilde{i} = i_1, \dots, i_m$, $\widetilde{n} = n_1, \dots, n_m$, and it is defined under replications $!^{i_1 \leq n_1} \dots !^{i_m \leq n_m}$. (There may be other instructions between these replications.) We define that a context has replications $!^{\widetilde{i} \leq \widetilde{n}}$ above the hole in a similar way. When $\widetilde{n} = n_1, \ldots, n_m$, we define $\prod \widetilde{n} = n_1 \times \cdots \times n_m$.

Definition 2 (Secrecy)**.** *Let x and V be such that $x \notin V$. Suppose that the variable x has type T and is defined under replications $!^{\widetilde{i} \leq \widetilde{n}}$ in Q. Let*

$$\begin{aligned}
R_x = {} & c_{s0}(); \textbf{new } b : \mathit{bool}; \overline{c_{s0}}\langle\rangle; \\
& \quad (!^{i_s \leq n_s} c_s[i_s](\widetilde{u} \leq \widetilde{n}); \textbf{if defined}(x[\widetilde{u}]) \textbf{ then} \\
& \quad\ \textbf{if } b \textbf{ then } \overline{c_s[i_s]}\langle x[\widetilde{u}]\rangle \textbf{ else} \\
& \quad\ \textbf{find } u'_s = i'_s \leq n_s \textbf{ suchthat defined}(y[i'_s], \widetilde{u}[i'_s]) \wedge \widetilde{u}[i'_s] = \widetilde{u} \textbf{ then } \overline{c_s[i_s]}\langle y[u'_s]\rangle \textbf{ else} \\
& \quad\ \textbf{new } y : T; \overline{c_s[i_s]}\langle y\rangle \\
& \quad \mid c'_s(b' : \mathit{bool}); \textbf{if } b = b' \textbf{ then event_abort } \mathsf{S} \textbf{ else event_abort } \overline{\mathsf{S}})
\end{aligned}$$

where the channels c_{s0}, c_s, c'_s, the variables $\widetilde{u}, u'_s, y, b, b'$, and the events S, $\overline{\mathsf{S}}$ do not occur in Q.

The process Q preserves the secrecy of *x with public variables V up to probability p when, for all evaluation contexts C acceptable for $Q \mid R_x$ with public variables V that do not contain S nor $\overline{\mathsf{S}}$, $\Pr[C[Q \mid R_x] : \mathsf{S}] - \Pr[C[Q \mid R_x] : \overline{\mathsf{S}}] \leq p(C)$.*

The process R_x chooses a random bit b, and then allows the adversary to query the variable x: if the adversary sends indices $\widetilde{u}$ on channel $c_s[i_s]$, and $x[\widetilde{u}]$ is defined, then the process R_x replies with the value of $x[\widetilde{u}]$ when b is true, and with a random value when b is false. The **find** in R_x makes sure that, if the indices $\widetilde{u}$ have already been queried, then the previous reply is sent; otherwise, a fresh random value y is chosen in the type T of x by **new** $y : T$, and sent as a reply. The replication $!^{i_s \leq n_s}$ in R_x allows the adversary to perform at most n_s such queries; n_s is chosen large enough so that it is not a limitation. Finally, the adversary sends on channel c'_s its guess b' for the bit b. If the guess is correct ($b' = b$), then the process R_x executes event S; otherwise, it executes event $\overline{\mathsf{S}}$. Intuitively, Q preserves the secrecy of x when the adversary cannot guess b, that is, it cannot distinguish whether the process outputs the value of the secret ($b = \mathsf{true}$) or outputs independent random numbers ($b = \mathsf{false}$).

Correspondences Correspondences [37] are properties of executed sequences of events, such as "if some event has been executed, then some other event has been executed". They are typically used for formalizing authentication. Given a correspondence *corr*, we define a distinguisher D such that $D(\mathcal{E}, a) = \mathsf{true}$ if and only if the sequence of events $\mathcal{E}$ satisfies the correspondence *corr*. We write this distinguisher simply *corr*, and write $\neg$*corr* for its negation.

Definition 3 (Correspondence)**.** *The process Q* satisfies the correspondence *corr with public variables V up to probability p if and only if, for all evaluation contexts C acceptable for Q with public variables V that do not contain events used in corr, $\Pr[C[Q] : \neg corr] \leq p(C)$.*

We refer the reader to [11] for more details on the verification of correspondences in CryptoVerif. We have:

Lemma 1. *If Q preserves the secrecy of x with public variables V up to probability p and C is an acceptable evaluation context for Q with public variables V, then for all $V' \subseteq V \cup \mathsf{var}(C)$, $C[Q]$ preserves the secrecy of x with public variables V' up to probability p' such that $p'(C') = p(C'[C])$.*

If Q satisfies a correspondence corr with public variables V up to probability p and C is an acceptable evaluation context for Q with public variables V that does not contain events used in corr, then for all $V' \subseteq V \cup \mathsf{var}(C)$, $C[Q]$ satisfies corr with public variables V' up to probability p' such that $p'(C') = p(C'[C])$.

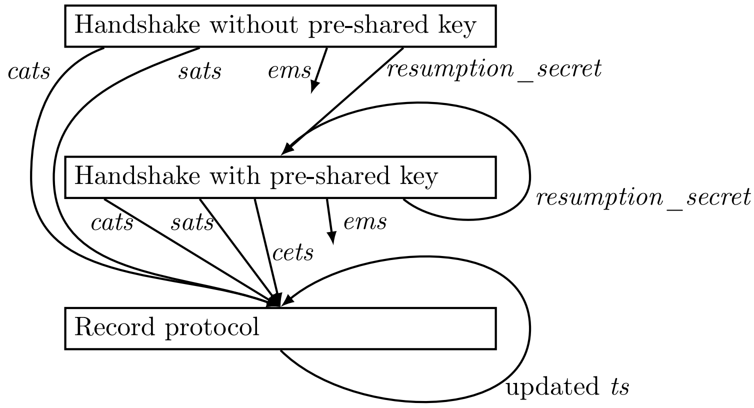


Figure 1: Structure of the composition

If $Q \approx_p^{V\cup\{x\}} Q'$ and Q preserves the secrecy of x with public variables V up to probability p', then Q' preserves the secrecy of x with public variables V up to probability p'' such that $p''(C) = p'(C) + 2 \times p(C[[\,] \mid R_x], t_{\mathsf{S}})$.

If $Q \approx_p^{V} Q'$ and Q satisfies a correspondence corr with public variables V up to probability p', then Q' satisfies corr with public variables V up to probability p'' such that $p''(C) = p'(C) + p(C, t_{corr})$.

Tables CryptoVerif also supports tables. Tables are lists of tuples shared between all honest participants of the protocol. The construct **insert** $Tbl(M_1, \dots, M_n); P$ inserts element $(M_1, \dots, M_n)$ in table Tbl, then runs P. The construct **get** $Tbl(x_1, \dots, x_l)$ **suchthat** M **in** P **else** P' tries to retrieve an element $(x_1, \dots, x_l)$ in the table Tbl such that M is true. When such an element is found, it executes P with $(x_1, \dots, x_l)$ bound to that element. When no such element is found, it executes P'. Equality tests $= M_i$ are also allowed instead of variables x_i; in this case, the table element must contain the value of M_i at the i-th position.

3 Structure of the proof of TLS 1.3

Our proof of TLS 1.3 relies on a previous analysis of the pieces that we compose [9,10]. Figure 1 summarizes the structure of the composition. We provide a brief sketch of those previous results here; more details are given in Appendix B.1.

The initial handshake, without pre-shared key, provides 4 keys at the end of the protocol: the client traffic secret *cats*, used by the record protocol for messages from the client to the server; the server traffic secret *sats*, used by the record protocol for messages from the server to the client; the exporter master secret *ems*, used to compute exporters (secrets generated by TLS that can be used by applications or other protocols); and the resumption secret *resumption_secret*, used as pre-shared key in the next handshake. For all these keys, CryptoVerif proves in particular secrecy, forward secrecy (with respect to the compromise of long-term client and server keys), and authentication.

In this model, the adversary has access to oracles that allow him to compromise the long-term client and server keys. The security properties are proved provided the long-term key of the peer is not compromised yet at the end of the handshake. As explained in the definition of secrecy (Section 2), this model does not include reveal queries for session keys; instead, CryptoVerif proves that all keys of the various sessions are indistinguishable from independent random keys, which is a stronger model [1].

The handshake with pre-shared key uses a pre-shared key and provides the same keys as above, with the same security properties. Additionally, it provides a client early traffic secret *cets*, computed after the first message of the protocol (`ClientHello`). The record protocol uses this traffic secret to send messages from the client to the server immediately after the `ClientHello` message, so-called 0-RTT data. The `ClientHello` message may be replayed and the server may also accept an altered `ClientHello` message, so CryptoVerif proves weaker properties about *cets*. When the `ClientHello` message is not altered, it proves in particular secrecy and non-injective authentication, since replays are possible. When the `ClientHello` message is altered, it essentially proves that the server has a value of *cets* that no one else has. In this case, the goal is to show that the record protocol that uses this value of *cets* never accepts messages.

Due to limitations of CryptoVerif, we cannot prove forward secrecy with respect to the compromise of the pre-shared key in the case of a handshake with pre-shared key and Diffie-Hellman key exchange. Hence, all properties that we prove for the handshake with pre-shared key rely on the secrecy of the pre-shared key. In the analysis of this part of the protocol, we can then consider that the long-term signature keys of the client and the server are compromised, and let the adversary deal with certificates and signatures if they appear. Therefore, the only common secret between our models of the initial handshake and of the handshake with pre-shared key is the pre-shared key. Furthermore, the analysis of the initial handshake allows the compromise of these long-term signature keys at the end of the handshake, so the security properties that we prove for the initial handshake remain valid.

Finally, the record protocol uses a traffic secret to derive an updated traffic secret, used for key updates, and a key and an initialization vector, used for encrypting and decrypting messages with an authenticated encryption scheme. CryptoVerif proves secrecy of the updated traffic secret, injective message authentication, and message secrecy. (The adversary cannot distinguish which one of two sets of messages is encrypted, similarly to the property we mentioned for S_1^b in Example 1.) We also consider two variants of the record protocol for 0-RTT. In the first variant, the receiver is replicated, so we have non-injective message authentication instead of the injective one. This variant is useful to support replays of unaltered `ClientHello` messages. In the second variant, the sender is additionally removed, and we show that the receiver never accepts a message. This variant is useful for altered `ClientHello` messages. The only common secret between the handshakes and the record protocol is the traffic secret.

The goal of our case study is to combine all these results in order to obtain security results for the full TLS 1.3 protocol.

4 The Most Basic Composition Theorem

As a warm-up, we present a very simple composition theorem, explained below.

Theorem 1. *Let C be any context with one hole, without replications above the hole and without* **event_abort**. *Let Q_1 be a process without* **event_abort**. *Let M be a term of type T. Let*

$$S_1 = C[\textbf{let } k = M \textbf{ in } \overline{c_1}\langle\rangle; Q_1]$$
$$S_2 = c_2(); \textbf{new } k : T; \overline{c_3}\langle\rangle; Q_2$$

where c_1, c_2, c_3 do not occur elsewhere in S_1, S_2; k is the only variable common to S_1 and S_2; S_1 and S_2 have no common channel, no common event, and no common table; and k does not occur in C and Q_1. Let c_1' be a fresh channel. Let

$$S_{composed} = C[\textbf{let } k = M \textbf{ in } \overline{c_1'}\langle\rangle; (Q_1 \mid Q_2)]$$

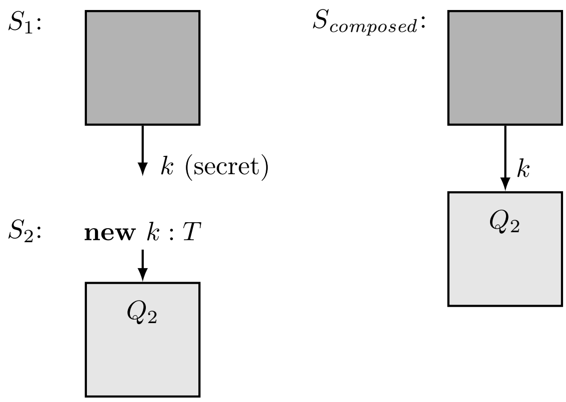


Figure 2: Illustration of Theorem 1

Let $S^\circ_{composed}$ be obtained from $S_{composed}$ by removing all events of S_1.

1. *If S_1 preserves the secrecy of k with public variables V ($k \notin V$) up to probability p, then there exists an evaluation context C' such that, for any $V_1 \subseteq V \cup (\mathsf{var}(S_1) \setminus \{k\})$, we have $S^\circ_{composed} \approx^{V_1}_{p'} C'[S_2]$ and C' is acceptable for S_2 without public variables, contains no event, runs in time at most $t_C + t_{Q_1}$, and does not alter the other parameters (replication bounds, lengths of bitstrings), where $p'(C_1, t_D) = p(C'_1)$ and C'_1 runs in time at most $t_{C_1} + t_{Q_2} + t_D$ and its other parameters are the same as those of C_1.*

2. *There exists an evaluation context C'' such that, for any $V' \subseteq \mathsf{var}(S_{composed})$, we have $S_{composed} \approx^{V'}_0 C''[S_1]$ and C'' is acceptable for S_1 with public variable k, contains the events of S_2, runs in time at most t_{Q_2}, and does not alter the other parameters.*

Moreover, C' is independent of Q_2 and C'' is independent of C and Q_1.

In this theorem, illustrated in Figure 2, we compose a system S_1 that establishes a key k with a system S_2 that runs Q_2 using a fresh random key k. The composed system runs S_1 and Q_2 using the key k provided by S_1. (The letters Q and S both represent CryptoVerif processes, in the same grammar. We use S for the systems that we compose and for the composed system, and Q for other processes.) Intuitively, the composition works because the secrecy of k allows us to replace k with a fresh random key. (An adversary cannot distinguish k from a fresh random key.) A detailed proof is provided in Appendix A.4. In contrast to the theorems of Section 5, in this theorem, S_1 is not a key exchange protocol: a single participant establishes the key k, so the composition is a lot easier.

The assumption that S_1 does not contain **event_abort** is useful because, in the definition of secrecy, when S_1 aborts before a message is sent on c_s, neither S nor $\overline{\mathsf{S}}$ is executed, so the adversary gets no advantage against the secrecy of k for these traces. However, these traces could still leak information on k that would break the composition. So we prevent S_1 from aborting. This is not a limitation in practice, because **event_abort** is typically introduced during security proofs, using Shoup's lemma [13,36], but does not occur in the initial protocol model.

The assumption that c_1, c_2, c_3 do not occur elsewhere in S_1, S_2 guarantees that messages sent to channel c_2 (resp. received from c_1, c_3) really go to the input (resp. come from the output) shown in the definitions of S_1 and S_2. The assumption that k does not occur in C and Q

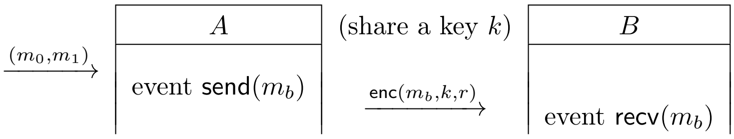


Figure 3: A picture of system S_2^b

guarantees that S_1 defines k but does not use it. The other assumptions on S_1 and S_2 can easily be obtained by renaming if necessary.

The first conclusion of Theorem 1, $S^\circ_{composed} \approx^{V_1}_{p'} C'[S_2]$, allows us to transfer security properties from S_2 to the composed system $S_{composed}$ using Lemma 1. In this property, we need to remove the events of S_1, because events can leak information on k even when S_1 preserves the secrecy of k according to Definition 2.

Similarly, the second conclusion of Theorem 1, $S_{composed} \approx^{V'}_0 C''[S_1]$, allows us to transfer security properties from S_1 to the composed system $S_{composed}$, provided these properties are proved with public variable k, because C'' uses k. These properties may allow us to compose again $S_{composed}$ with another protocol.

In our TLS case study, we use this composition theorem to deal with key updates in the record protocol. The system S_1 runs the record protocol and computes an updated traffic secret from a traffic secret. This updated traffic secret is the key k in the composition theorem. The system S_2 uses this key k to run the record protocol again. The composition theorem allows us to obtain security properties for a record protocol that performs a key update. We compose again recursively to allow any number of key updates. The next example presents a simplified version of this situation, to illustrate the theorem more formally.

Example 1. *Consider the system S_2^b defined by*

$$\begin{aligned} S_2^b = {} & c_2(); \textbf{new } k : T; \overline{c_3}\langle\rangle; \\ & (c_4((m_0 : T_m, m_1 : T_m)); \textbf{new } r : T_r; \textbf{event } \mathsf{send}(m_b); \overline{c_5}\langle \mathsf{enc}(m_b, k, r)\rangle \\ & \mid c_6(y : \mathit{bitstring}); \textbf{let } \mathsf{i}_\perp(m) = \mathsf{dec}(y, k) \textbf{ in event } \mathsf{recv}(m)) \end{aligned}$$

where all bitstrings in T_m have the same length. This system is illustrated in Figure 3. The system S_2^b chooses a key k, and then runs two participants, say A and B, in parallel. When A receives two messages m_0, m_1 of the same length on channel c_4, it sends the encryption of m_b under k on channel c_5 and records this emission with the event $\mathsf{send}(m_b)$. *When B receives a ciphertext on channel c_6, it decrypts that ciphertext, stores the plaintext in m, and executes event* $\mathsf{recv}(m)$. *(The decryption function* dec *returns $\perp$ when it fails, and the function $\mathsf{i}_\perp$ is the natural injection from bitstring to bitstring $\cup \{\perp\}$, so that the equality $\mathsf{i}_\perp(m) = \mathsf{dec}(y, k)$ holds when the decryption succeeds and m is the corresponding cleartext.) When* $(\mathsf{enc}, \mathsf{dec})$ *is an authenticated encryption scheme, we have $S_2^0 \approx_{p_1} S_2^1$, which means that the adversary can distinguish whether m_0 or m_1 was encrypted with probability at most p_1, and for $b \in \{0, 1\}$, S_2^b satisfies the correspondence*

$$\mathit{corr} = \textbf{inj-event}(\mathsf{recv}(m)) \Longrightarrow \textbf{inj-event}(\mathsf{send}(m)) \quad (1)$$

up to probability p_2 without public variables, which means that each execution of event $\mathsf{recv}(m)$ *is preceded by a distinct execution of event* $\mathsf{send}(m)$*, up to cases of probability at most p_2. (The probabilities p_1 and p_2 come from the probabilities of breaking the security properties of the* *encryption scheme.) By composing S_1 with S_2^b, we obtain*

$$S^b_{composed} = C[\textbf{let } k = M \textbf{ in } \overline{c'_1}\langle\rangle; (Q_1 \\ \mid c_4((m_0 : T_m, m_1 : T_m)); \textbf{new } r : T_r; \textbf{event } \mathsf{send}(m_b); \overline{c_5}\langle\mathsf{enc}(m_b, k, r)\rangle \\ \mid c_6(y : \mathit{bitstring}); \textbf{let } \mathsf{i}_\perp(m) = \mathsf{dec}(y, k) \textbf{ in event } \mathsf{recv}(m))]$$

Let $S^{b,\circ}_{composed}$ be obtained from $S^b_{composed}$ by removing all events of S_1. Let $V_1 = \mathsf{var}(S_1) \setminus \{k\}$. By Theorem 1, we have $S^{b,\circ}_{composed} \approx^{V_1}_{p'} C'[S^b_2]$ for $b \in \{0,1\}$. (The context C' does not depend on b because C' is independent of Q_2 in Theorem 1.) By Lemma 1, $C'[S^b_2]$ satisfies the correspondence (1) *up to probability p'_2 with public variables V_1, where $p'_2(C_1) = p_2(C_1[C'])$, and so $S^{b,\circ}_{composed}$ satisfies* (1) *up to probability p''_2, where $p''_2(C_1) = p'(C_1, t_{corr}) + p'_2(C_1) = p(C'_1) + p_2(C''_1)$, C'_1 runs in time at most $t_{C_1} + t_{\mathsf{enc}} + t_{\mathsf{dec}} + t_{corr}$, $C''_1 = C_1[C']$ runs in time at most $t_{C_1} + t_{C'} = t_{C_1} + t_C + t_{Q_1}$, and their other parameters are the same as those of C_1. (The other parameters of $C''_1 = C_1[C']$ are the same as those of C_1 because C' does not alter these parameters.) Therefore, $S^b_{composed}$ also satisfies* (1) *up to probability p''_2, since S_1 does not contain the events* send *and* recv*. Moreover, assuming S_1 does not contain events, we have $S^0_{composed} = S^{0,\circ}_{composed} \approx^{V_1}_{p'} C'[S^0_2] \approx^{V_1}_{p'_1} C'[S^1_2] \approx^{V_1}_{p'} S^{1,\circ}_{composed} = S^1_{composed}$ where $p'_1(C_1, t_D) = p_1(C_1[C'], t_D)$, so by transitivity, $S^0_{composed} \approx^{V_1}_{2p'+p'_1} S^1_{composed}$: in the composed system, the adversary can distinguish whether m_0 or m_1 was encrypted with probability at most $2p' + p'_1$.*

5 Main Composition Results

This section presents our main composition theorems. We first need to introduce preliminary notions and lemmas.

5.1 Transferring Security Properties

We first generalize the notion of indistinguishability. The more general notion still allows us to transfer security properties from a process to another, as indistinguishability does by Lemma 1.

Definition 4. *We write $Q \overset{\sim}{\rightarrow}^{V,V'}_{f,p} Q'$ if, and only if, for all evaluation contexts C acceptable for Q with public variables V and all distinguishers D that run in time at most t_D, $C' = f(C)$ is an evaluation context acceptable for Q' with public variables V' such that $|\Pr[C[Q] : D] - \Pr[C'[Q'] : D]| \leq p(C, t_D)$.*

Intuitively, $Q \overset{\sim}{\rightarrow}^{V,V'}_{f,p} Q'$ means that, for each adversary against Q (represented by the context C), there exists a modified adversary against Q' (represented by the context $C' = f(C)$) such that $C[Q]$ and $C'[Q']$ behave similarly. (The difference between the probabilities $\Pr[C[Q] : D]$ and $\Pr[C'[Q'] : D]$ is at most $p(C, t_D)$.)

Indistinguishability corresponds to the particular case in which f is the identity: $f(C) = C$. Being able to transform the context C by the function f is useful in composition proofs, in particular because the variables are not always numbered in the same way in the symmetric key protocol and in the composed system. In this case, f performs the renumbering of the variables.

The rest of this section shows that $Q \overset{\sim}{\rightarrow}^{V,V'}_{f,p} Q'$ allows us to transfer indistinguishability, correspondence, and secrecy properties from Q' to Q.

Lemma 2. *If $Q'_1 \approx^{V'}_{p'} Q'_2$, $Q_1 \mathrel{\overset{\sim}{\rightarrow}}^{V,V'}_{f,p_1} Q'_1$, and $Q_2 \mathrel{\overset{\sim}{\rightarrow}}^{V,V'}_{f,p_2} Q'_2$, then $Q_1 \approx^{V}_{p''} Q_2$, where $p''(C, t_D) = p_1(C, t_D) + p'(f(C), t_D) + p_2(C, t_D)$.*

Intuitively, if there is an adversary (represented by the context C), that can distinguish Q_1 from Q_2 with probability p'', then the properties $Q_1 \mathrel{\overset{\sim}{\rightarrow}}^{V,V'}_{f,p_1} Q'_1$ and $Q_2 \mathrel{\overset{\sim}{\rightarrow}}^{V,V'}_{f,p_2} Q'_2$ guarantee that there is a modified adversary (represented by the context $C' = f(C)$) that can distinguish Q'_1 from Q'_2 with probability at least $p''(C, t_D) - p_1(C, t_D) - p_2(C, t_D)$. Since $Q'_1 \approx^{V'}_{p'} Q'_2$, this probability is at most $p'(f(C), t_D)$, so we obtain Lemma 2. Lemma 3 is a similar result for correspondences.

Lemma 3. *If Q' satisfies a correspondence corr with public variables V' up to probability p' and $Q \mathrel{\overset{\sim}{\rightarrow}}^{V,V'}_{f,p} Q'$, where f is such that when C does not contain events used by corr, neither does $f(C)$, then Q satisfies corr with public variables V up to probability p'', where $p''(C) = p(C, t_{corr}) + p'(f(C))$.*

Definition 5. *Assuming $Q \mathrel{\overset{\sim}{\rightarrow}}^{V,V'}_{f,p} Q'$, we say that f is* secrecy-preserving *for $x' \mapsto (x, f_{\mathrm{sec}})$ when we have: If Q' preserves the secrecy of x' with public variables $V' \setminus \{x'\}$ up to probability p', $x' \in V'$, and $x \in V$, then Q preserves the secrecy of x with public variables $V \setminus \{x\}$ up to probability p'', where $p''(C_0) = 2p(C_0[[] \mid R_x], t_{\mathsf{S}}) + p'(f_{\mathrm{sec}}(C_0))$.*

Definition 5 just defines that function f allows us to transfer secrecy properties. This property holds in particular when, for every evaluation context C_0 acceptable for $Q \mid R_x$ with public variables $V \setminus \{x\}$, there exist C'_0 and C''_0 such that $f(C_0[[] \mid R_x]) = C'_0[C''_0[] \mid R_{x'}]$. This condition guarantees that f preserves the form of contexts that we use to test secrecy $C_0[[] \mid R_x]$, just allowing the addition of a context C''_0 before the secrecy test; this addition preserves secrecy by Lemma 1. (This result is detailed in Lemma 10 in Appendix A.5.) In our composition proofs, we use this condition, as well as others detailed in the proofs themselves.

5.2 Hash Oracles

The systems S_1 and S_2 that we compose may use hash oracles. In this paper, we consider only non-programmable random oracles. The systems S_1 and S_2 may share the same hash oracles, which appear once in the composed system. To allow the sharing of oracles between S_1 and S_2, we must treat these oracles specially. In this section, we introduce notations and a lemma that allow us to do that.

We assume that there are L hash oracles ($L \geq 1$), and use the following notations: for each $l \leq L$, h_l is a function of type $T_{hk_{\mathsf{h},l}} \times T_{\mathsf{h},l} \to T'_{\mathsf{h},l}$,

$$
\begin{aligned}
Q_{\mathsf{h}} &= \prod\nolimits_{l=1}^{L} !^{i_{\mathsf{h},l} \leq n_{\mathsf{h},l}} c_{\mathsf{h}3,l}[i_{\mathsf{h},l}](x_{\mathsf{h},l} : T_{\mathsf{h},l}); \overline{c_{\mathsf{h}4,l}[i_{\mathsf{h},l}]}\langle \mathsf{h}_l(hk_{\mathsf{h},l}, x_{\mathsf{h},l})\rangle \\
C_{\mathsf{h}} &= c_{\mathsf{h}1}(); \mathbf{new}\ hk_{\mathsf{h},1} : T_{hk_{\mathsf{h},1}}; \ldots \mathbf{new}\ hk_{\mathsf{h},L} : T_{hk_{\mathsf{h},L}}; \overline{c_{\mathsf{h}2}}\langle\rangle; ([] \mid Q_{\mathsf{h}})
\end{aligned}
$$

The context C_{h} first chooses the keys $hk_{\mathsf{h},l}$ ($l \leq L$). This choice models the choice of the random oracles themselves. It is triggered by the reception of a message on $c_{\mathsf{h}1}$ and followed by an output on $c_{\mathsf{h}2}$. Then, C_{h} runs the process Q_{h} in parallel with the hole. The process Q_{h} represents L hash oracles: the l-th hash oracle can be called at most $n_{\mathsf{h},l}$ times; it receives its argument $x_{\mathsf{h},l}$ on channel $c_{\mathsf{h}3,l}[i_{\mathsf{h},l}]$ ($i_{\mathsf{h},l} \leq n_{\mathsf{h},l}$) and returns the hash of $x_{\mathsf{h},l}$ on channel $c_{\mathsf{h}4,l}[i_{\mathsf{h},l}]$. We use $\prod_{l=1}^{L} Q_l$ to denote the parallel composition $Q_1 \mid \ldots \mid Q_L$. The context C_{h} is not an evaluation context (because it always chooses the keys $hk_{\mathsf{h},l}$ before running the process in the hole). Let Q'_{h} and C'_{h} be obtained from Q_{h} and C_{h} by renaming the replication bounds $n_{\mathsf{h},l}$ into $n'_{\mathsf{h},l}$ and the channels

$c_{\mathsf{h}1}$, $c_{\mathsf{h}2}$, $c_{\mathsf{h}3,l}$, $c_{\mathsf{h}4,l}$ into $c'_{\mathsf{h}1}$, $c'_{\mathsf{h}2}$, $c'_{\mathsf{h}3,l}$, $c'_{\mathsf{h}4,l}$ respectively. Similarly, let Q''_{h} and C''_{h} be obtained from Q_{h} and C_{h} by renaming the replication bounds $n_{\mathsf{h},l}$ into $n''_{\mathsf{h},l}$ and the channels $c_{\mathsf{h}1}$, $c_{\mathsf{h}2}$, $c_{\mathsf{h}3,l}$, $c_{\mathsf{h}4,l}$ into $c''_{\mathsf{h}1}$, $c''_{\mathsf{h}2}$, $c''_{\mathsf{h}3,l}$, $c''_{\mathsf{h}4,l}$ respectively. We say that a process is *hash-well-formed* when, for all $l \leq L$, it uses $hk_{\mathsf{h},l}$ only in terms of the form $\mathsf{h}_l(hk_{\mathsf{h},l}, M)$ for some term M, it does not use the channels $c_{\mathsf{h}1}$, $c_{\mathsf{h}2}$, $c_{\mathsf{h}3,l}$, $c_{\mathsf{h}4,l}$, $c'_{\mathsf{h}1}$, $c'_{\mathsf{h}2}$, $c'_{\mathsf{h}3,l}$, $c'_{\mathsf{h}4,l}$, $c''_{\mathsf{h}1}$, $c''_{\mathsf{h}2}$, $c''_{\mathsf{h}3,l}$, $c''_{\mathsf{h}4,l}$, and it does not use the variables $x_{\mathsf{h},l}$.

In the particular case in which there is no hash oracle ($L = 0$), we define $C_{\mathsf{h}} = C'_{\mathsf{h}} = C''_{\mathsf{h}} = [\,]$, the empty context.

Given a process Q, we write $n_{\mathsf{h},l,Q}$ for the maximum number of evaluations of $\mathsf{h}_l(hk_{\mathsf{h},l}, \dots)$ in Q. The same notation applies to contexts C and terms M.

The next lemma is the main technical tool that we use to deal with hash oracles. It allows us to move the hash oracles under an evaluation context.

Lemma 4. *If C is an evaluation context and $C[Q]$ is hash-well-formed, then there exists an evaluation context C' such that for all V such that $V \cap \mathsf{var}(C_{\mathsf{h}}) = \emptyset$,*

$$C_{\mathsf{h}}[C[Q]] \approx_0^V C'[C'_{\mathsf{h}}[Q]]$$

where the context C' is independent of Q, runs in time at most t_C, and for all $l \leq L$, C' calls the l-th hash oracle in C'_{h} at most $n_{\mathsf{h},l,C}$ times, so $n'_{\mathsf{h},l} = n_{\mathsf{h},l} + n_{\mathsf{h},l,C}$. (The symbol $n_{\mathsf{h},l}$ occur in C_{h} and $n'_{\mathsf{h},l}$ occurs in C'_{h}.) The other parameters of C' are the same as those of C.

In this lemma, the context C directly calls the hash functions h_l, while the context C' performs the same hash evaluations by calling the hash oracles defined by Q'_{h} inside C'_{h}. (The context C' cannot call h_l directly, because it does not have access to the keys $hk_{\mathsf{h},l}$. The context C cannot call the hash oracles of C_{h} because it is hash-well-formed, so it does not use the channels $c_{\mathsf{h}3,l}$ and $c_{\mathsf{h}4,l}$.)

5.3 Replication

When we compose a key exchange protocol S_1 with a protocol S_2 that uses the key, we typically run n sessions of the key exchange, and each session produces a fresh key. Therefore, we need to consider n independent sessions of S_2, each with a different fresh key. In this section, we show how to infer security properties (indistinguishability, secrecy, correspondences) of a protocol that runs n independent sessions from the properties of a protocol that runs a single session. (In the vocabulary of [16], we consider that the protocol that uses the key is *single-session reducible*, and we obtain results similar to theirs for the reduction to a single session [16, Appendix B], but in the context of CryptoVerif.)

Let us consider a protocol Q that runs a single session. We can model a protocol that runs n sessions of Q by adding a replication at the top of Q: $!^{i \leq n} Q$. Then all variables defined in Q implicitly have one more index, i, because they are defined under $!^{i \leq n}$. That allows us to distinguish the variables used in different sessions. However, this is not sufficient: we want the adversary to be able to know to (resp. from) which session it sends (resp. receives) messages, so we add the replication index i to the channels of inputs and outputs in Q. Similarly, we can add the replication index i as argument of events in Q, to be able to relate events that belong to the same session. Considering the process S_2^b of Example 1, that yields:

$$\begin{aligned}
&!^{i \leq n} c_2[i](); \mathbf{new}\ k : T; \overline{c_3[i]}\langle\rangle;\\
&\quad (c_4[i]((m_0 : T_m, m_1 : T_m)); \mathbf{new}\ r : T_r; \mathbf{event}\ \mathsf{send}(i, m_b); \overline{c_5[i]}\langle \mathsf{enc}(m_b, k, r)\rangle\\
&\quad \mid c_6[i](y : \mathit{bitstring}); \mathbf{let}\ \mathsf{i}_\perp(m) = \mathsf{dec}(y, k)\ \mathbf{in\ event}\ \mathsf{recv}(i, m))
\end{aligned}$$

and this process satisfies the correspondence

$$\textbf{inj-event}(\mathsf{recv}(i,m)) \Longrightarrow \textbf{inj-event}(\mathsf{send}(i,m))$$

that is, each execution of $\mathsf{recv}(i,m)$ is preceded by a distinct execution of $\mathsf{send}(i,m)$, up to cases of negligible probability. In this process, partnered sessions (which use the same key k) have the same replication index i. However, this property is not preserved by composition: in a key exchange protocol, partnered sessions are typically the ones that exchange the same messages, and they do not necessarily have the same replication index. This will also be true in the composed system. Partnered sessions can then be determined by a *session identifier* computed from the messages exchanged in the protocol, as in [1, 7, 16, 19]: partnered sessions have the same session identifier. In the composition, the session identifier will be determined by the key exchange protocol. Therefore, we consider that the protocol that uses the key receives the session identifier in a variable x, as follows:

$$\begin{aligned}
&!^{i\leq n} c_2[i](x : T_{\mathsf{sid}}); \textbf{new } k : T; \overline{c_3[i]}\langle\rangle;\\
&\quad (c_4[i]((m_0 : T_m, m_1 : T_m)); \textbf{new } r : T_r; \textbf{event } \mathsf{send}(x, m_b); \overline{c_5[i]}\langle \mathsf{enc}(m_b, k, r)\rangle\\
&\quad \mid c_6[i](y : \mathit{bitstring}); \textbf{let } \mathsf{i}_\perp(m) = \mathsf{dec}(y, k) \textbf{ in event } \mathsf{recv}(x, m))
\end{aligned}$$

We use the session identifier x instead of the replication index i in events. The only missing ingredient in the above process is that the same session identifier should never be used twice, to avoid confusions between several sessions. The **find** construct allows us to verify that, by comparing x to the previously received session identifiers. This explanation leads us to the following definition:

Definition 6. *Given a process P, and replication indices $\widetilde{i}$ and a variable x that do not occur in P, we write $\mathsf{AddIdxSid}(\widetilde{i} \leq \widetilde{n}, x : T_{\mathsf{sid}}, P)$ for the process obtained by adding indices $\widetilde{i}$ at the beginning of each sequence of indices of channels in inputs and outputs and at the beginning of the indices of each variable defined in P (implicit when current replication indices are omitted), adding variable x at the beginning of each event and at the beginning of each insertion in a table, and adding the test $= x$ at the beginning of each* **get** *in a table.*

Given a correspondence corr, we write $\mathsf{AddSid}(T_{\mathsf{sid}}, \mathit{corr})$ for the correspondence obtained by choosing a fresh variable x of type T_{sid} and adding it at the beginning of each event in corr.

When Q is of the form $Q = c(); P$ and the channels c and c' and the replication indices $\widetilde{i}$ do not occur in P, we define

$$\begin{aligned}
\mathsf{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c', T_{\mathsf{sid}}, Q) = {} & !^{\widetilde{i}\leq\widetilde{n}} c'[\widetilde{i}](x : T_{\mathsf{sid}});\\
& \textbf{find } \widetilde{u} = \widetilde{i'} \leq \widetilde{n} \textbf{ suchthat defined}(x[\widetilde{i'}], x'[\widetilde{i'}]) \wedge x = x[\widetilde{i'}] \textbf{ then yield else}\\
& \textbf{let } x' = \mathsf{cst} \textbf{ in } \mathsf{AddIdxSid}(\widetilde{i} \leq \widetilde{n}, x : T_{\mathsf{sid}}, P)
\end{aligned}$$

where x, x', and $\widetilde{u}$ are fresh variables.

The process $\mathsf{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c', T_{\mathsf{sid}}, Q)$ is the replicated version of process $Q = c(); P$: it corresponds to $\widetilde{n}$ copies of Q indexed by $\widetilde{i} \leq \widetilde{n}$, as shown by the replication $!^{\widetilde{i}\leq\widetilde{n}}$. However, it additionally manages the session identifier and replication indices as detailed in the explanation above. The first input in $\mathsf{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c', T_{\mathsf{sid}}, Q)$ receives the session identifier x, the subsequent **find** checks that the same x is never used twice, so that there is a bijection between the value of x and the replication indices $\widetilde{i}$. (When the received session identifier x is equal to a previous one $x[\widetilde{i'}]$ with which P was run, it just executes **yield**, which returns control to the adversary. We record that P is run in session $\widetilde{i}$ by defining the variable $x'[\widetilde{i}]$ as a constant value cst. The **find** requires that $x'[\widetilde{i'}]$ be defined, which means that P was run in session $\widetilde{i'}$. In particular, $\widetilde{i'} \neq \widetilde{i}$, because $x'[\widetilde{i}]$ is not defined yet when the **find** is executed.) Finally, the process P that follows the input is executed, with the appropriate additions of the replication indices $\widetilde{i}$ or the session identifier x to channels, variables, events, and tables, as defined by $\mathsf{AddIdxSid}(\widetilde{i \leq n}, x : T_{\mathsf{sid}}, P)$.

Lemmas 5 and 6 below show that indistinguishability, secrecy, and correspondence properties are preserved by adding a replication. The hash oracles, when present, are left outside the replication.

Lemma 5. *Suppose that $V \cap \mathsf{var}(C'_{\mathsf{h}}) = \emptyset$, $Q = c(); P$, $Q' = c(); P'$, Q and Q' are hash-well-formed and do not contain events, $Q_! = \mathsf{AddReplSid}(\widetilde{i \leq n}, c', T_{\mathsf{sid}}, Q)$, and $Q'_! = \mathsf{AddReplSid}(\widetilde{i \leq n}, c', T_{\mathsf{sid}}, Q')$. If $C'_{\mathsf{h}}[Q] \approx^V_p C'_{\mathsf{h}}[Q']$, then $C_{\mathsf{h}}[Q_!] \approx^V_{p'} C_{\mathsf{h}}[Q'_!]$ where $p'(C, t_D) = \prod \widetilde{n} \times p(C', t_D)$ and the context C' runs in time at most $t_C + (\prod \widetilde{n} - 1) \times \max(t_Q, t_{Q'})$, calls the l-th hash oracle at most $n'_{\mathsf{h},l} = n_{\mathsf{h},l} + (\prod \widetilde{n} - 1) \times \max(n_{\mathsf{h},l,Q}, n_{\mathsf{h},l,Q'})$ times where C calls the l-th hash oracle at most $n_{\mathsf{h},l}$ times, and the other parameters of C' are the same as those of C.*

Lemma 6. *Suppose that $V \cap \mathsf{var}(C'_{\mathsf{h}}) = \emptyset$, Q is a hash-well-formed process, and $Q_! = \mathsf{AddReplSid}(\widetilde{i \leq n}, c', T_{\mathsf{sid}}, Q)$.*

1. *If $C'_{\mathsf{h}}[Q]$ preserves the secrecy of x with public variables V up to probability p with $x \notin \mathsf{var}(C'_{\mathsf{h}})$ and Q does not contain* **event_abort**, *then $C_{\mathsf{h}}[Q_!]$ preserves the secrecy of x with public variables V up to probability p'; and*

2. *if $C'_{\mathsf{h}}[Q]$ satisfies the correspondence corr with public variables V up to probability p, then $C_{\mathsf{h}}[Q_!]$ satisfies the correspondence $\mathsf{AddSid}(T_{\mathsf{sid}}, corr)$ with public variables V up to probability p';*

where $p'(C) = \prod \widetilde{n} \times p(C')$ and the context C' runs in time at most $t_C + (\prod \widetilde{n} - 1)t_Q$, calls the l-th hash oracle at most $n'_{\mathsf{h},l} = n_{\mathsf{h},l} + (\prod \widetilde{n} - 1)n_{\mathsf{h},l,Q}$ times where C calls the l-th hash oracle at most $n_{\mathsf{h},l}$ times, and the other parameters of C' are the same as those of C.

Example 2. *Letting $S^b_{2!} = \mathsf{AddReplSid}(i \leq n, c'_2, T_{\mathsf{sid}}, S^b_2)$, by Lemma 5, we obtain $S^0_{2!} \approx_{p'_1} S^1_{2!}$ and by Lemma 6, $S^b_{2!}$ satisfies the correspondence*

$$\textbf{inj-event}(\mathsf{recv}(x, m)) \Longrightarrow \textbf{inj-event}(\mathsf{send}(x, m))$$

up to probability p'_2 without public variables, where $p'_1(C, t_D) = n \times p_1(C', t_D)$, $p'_2(C) = n \times p_2(C')$, and C' runs in time at most $t_C + (n-1) \times t_{S^b_2}$ and its other parameters are the same as those of C. (Note that $t_{S^0_2} = t_{S^1_2}$.)

5.4 Main Composition Theorem

Finally, we obtain our main composition theorem. We write $P\{M/x\}$ for the process obtained from P by substituting M for x. We denote by $C + t_D$ a context that runs in time at most $t_C + t_D$ and such that the other parameters of $C + t_D$ are the same as those of C.

Theorem 2 (Main composition theorem)**.** *Let C be any context with two holes, with replications $!^{\widetilde{i} \leq \widetilde{n}}$ above the first hole and $!^{\widetilde{i'} \leq \widetilde{n'}}$ above the second hole and without* **event_abort**. *Let Q_{1A} and Q_{1B} be processes without* **event_abort**. *Let k, k_A, k_B be variables of type T. Let*

$$\begin{aligned} Q_1 = C[&\textbf{event } e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \textbf{let } k'_A = k_A \textbf{ in } \overline{c_A[\widetilde{i}]}\langle M_A\rangle; Q_{1A},\\ &\textbf{event } e_B(\mathsf{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\widetilde{i'}]}\langle M_B\rangle; Q_{1B}] \end{aligned}$$

$$Q_2 = c_1(); \textbf{new}\ k : T; \overline{c_2}\langle\rangle; (Q_{2A} \mid Q_{2B})$$
$$S_1 = C_{\mathsf{h}}[Q_1]$$
$$S_2 = C'_{\mathsf{h}}[\mathsf{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c'_1, T_{\mathsf{sid}}, Q_2)]$$

where Q_1 and Q_2 are hash-well-formed; $\widetilde{msg}_A$ is a sequence of variables defined in C above the first hole and input or output by C above the first hole or by the output $\overline{c_A[\widetilde{i}]}\langle M_A\rangle$; $\widetilde{msg}_B$ is a sequence of variables input or output by C above the second hole; sid *is a function that takes a sequence of messages and returns a session identifier of type T_{sid}; C, Q_{1A}, Q_{1B}, Q_{2A}, and Q_{2B} make all their inputs and outputs on pairwise distinct channels with indices the current replication indices; $c_A, c_B, c_1, c'_1, c_2, k'_A, e_A, e_B$ do not occur elsewhere in S_1, S_2; S_1 and S_2 have no common variable, no common channel, no common event, and no common table; S_1 and S_2 do not contain* **newChannel***; and there is no* **defined** *condition in Q_2.*

Let $Q'_{2A} = \mathsf{AddIdxSid}(\widetilde{i} \leq \widetilde{n}, x : T_{\mathsf{sid}}, Q_{2A})$ and $Q'_{2B} = \mathsf{AddIdxSid}(\widetilde{i'} \leq \widetilde{n}', x : T_{\mathsf{sid}}, Q_{2B})$. Let c'_A, c'_B be fresh channels. Let

$$\begin{aligned}
&Q_{composed} = \\
&\quad C[\textbf{event}\ e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \overline{c'_A[\widetilde{i}]}\langle M_A\rangle; (Q_{1A} \mid Q'_{2A}\{k_A/k, \mathsf{sid}(\widetilde{msg}_A)/x\}),\\
&\qquad \textbf{event}\ e_B(\mathsf{sid}(\widetilde{msg}_B), k_B); \overline{c'_B[\widetilde{i'}]}\langle M_B\rangle; (Q_{1B} \mid Q'_{2B}\{k_B/k, \mathsf{sid}(\widetilde{msg}_B)/x\})]\\
&S_{composed} = C''_{\mathsf{h}}[Q_{composed}]
\end{aligned}$$

Let $S^{\circ}_{composed}$ be obtained from $S_{composed}$ by removing all events of S_1.

Let $t_1 = t_C + \prod \widetilde{n} \times (t_{M_A} + t_{Q_{1A}}) + \prod \widetilde{n}' \times (t_{M_B} + t_{Q_{1B}})$ be an upper bound on the runtime of Q_1, $t_2 = \prod \widetilde{n} \times t_{Q_{2A}} + \prod \widetilde{n}' \times t_{Q_{2B}}$ be an upper bound on the runtime of Q'_{2A} and Q'_{2B} in $Q_{composed}$, $n_{\mathsf{h},l,1} = n_{\mathsf{h},l,C} + \prod \widetilde{n} \times (n_{\mathsf{h},l,M_A} + n_{\mathsf{h},l,Q_{1A}}) + \prod \widetilde{n}' \times (n_{\mathsf{h},l,M_B} + n_{\mathsf{h},l,Q_{1B}})$, and $n_{\mathsf{h},l,2} = \prod \widetilde{n} \times n_{\mathsf{h},l,Q_{2A}} + \prod \widetilde{n}' \times n_{\mathsf{h},l,Q_{2B}}$.

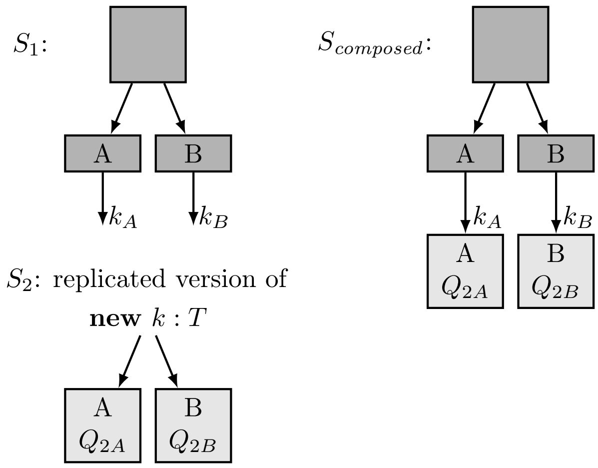
1. *If S_1 preserves the secrecy of k'_A with public variables V ($V \subseteq \mathsf{var}(S_1) \setminus (\{k_A, k'_A\} \cup \mathsf{var}(C_{\mathsf{h}}))$) up to probability p and satisfies the correspondences*

$$\textbf{inj-event}(e_B(sid, k)) \Longrightarrow \textbf{inj-event}(e_A(sid, k, \widetilde{i})) \tag{2}$$
$$\textbf{event}(e_A(sid, k_1, \widetilde{i}_1)) \wedge \textbf{event}(e_A(sid, k_2, \widetilde{i}_2)) \Longrightarrow \widetilde{i}_1 = \widetilde{i}_2 \tag{3}$$

   with public variables $V \cup \{k'_A\}$ up to probabilities p' and p'' respectively, then there exists f such that, for any $V_1 \subseteq V \cup (\mathsf{var}(Q_2) \setminus (\{k\} \cup \mathsf{var}(C'_{\mathsf{h}})))$, we have $S^{\circ}_{composed} \overset{\sim V_1,V_2}{\rightarrow}_{f,p_3} S_2$ where $V_2 = V_1 \cap \mathsf{var}(Q_2)$; $p_3(C_3, t_D) = p(C'_3 + t_D) + p'(C'_3, t_D) + p''(C'_3, t_D)$ and, assuming C_3 calls the l-th hash oracle $n''_{\mathsf{h},l}$ times, the context C'_3 runs in time at most $t_{C_3} + t_2$, calls the l-th hash oracle at most $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$ times, and its other parameters are the same as those of C_3; $f(C_3)$ contains the same events as C_3, runs in time at most $t_{C_3} + t_1$, calls the l-th hash oracle at most $n'_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,1}$ times, and its other parameters are the same as those of C_3; if $y \in V_2$, then f is secrecy-preserving for $y \mapsto (y, f_{\mathrm{sec}})$ where $f_{\mathrm{sec}}(C_3)$ has the same parameters as $f(C_3)$.

2. *There exists an evaluation context C'_4 such that, for any $V' \subseteq \mathsf{var}(S_{composed}) \setminus (\{k'_A\} \cup \mathsf{var}(C''_{\mathsf{h}}))$, we have $S_{composed} \approx_0^{V'} C'_4[S_1]$ and C'_4 is acceptable for S_1 with public variables k'_A, k_B, contains the events of S_2, runs in time at most t_2, calls the l-th hash oracle at most $n_{\mathsf{h},l,2}$ times, so $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$, and does not alter the other parameters.*

Moreover, f is independent of the details of Q_{2A} and Q_{2B}: it depends only on the channels of Q_{2B}, whether they are for input or for output, under which replications and with which type of data; C'_4 is independent of Q_{1A} and Q_{1B}.



(S_1 may run several sessions of A and B.)

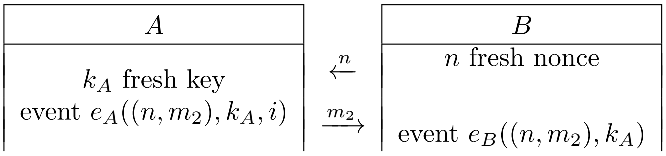
Figure 4: Illustration of Theorem 2

In this theorem, illustrated in Figure 4, the system S_1 is a key exchange protocol that provides a key to two participants: A executes event e_A and stores the key in k_A and k'_A, and B executes event e_B and stores the key in k_B. The system S_2 creates a fresh key, and also involves two participants: A executes Q_{2A} and B executes Q_{2B}. The composed system $S_{composed}$ combines S_1 and S_2 so that A executes Q_{2A} with the key k_A and B executes Q_{2B} with the key k_B, after the key exchange S_1 provides the key. These systems may share hash oracles, included in C_{h}, C'_{h}, and C''_{h}. (These contexts use the same hash functions. The hash oracles are omitted in Figure 4.)

This theorem requires the key exchange to satisfy the following security properties. It must guarantee the secrecy of the key obtained by A, k'_A, and injective authentication of A and B, as formalized by the correspondence (2). This correspondence means that each execution of event $e_B(\widetilde{msg}, k)$ is preceded by a distinct execution of event $e_A(\widetilde{msg}, k, \widetilde{i})$ for some $\widetilde{i}$, except in cases of probability at most p'. These two properties imply secrecy of the obtained key on B's side, since all keys that B has are also keys that A has. The correspondence (3) means that the event e_A is executed at most once for each session identifier *sid*, since all such executions must have the same replication indices $\widetilde{i}$. It allows us to use the session identifier *sid* as argument x to identify the session in the system S_2. It is easy to prove in practice, both using CryptoVerif and manually: it is sufficient to notice that *sid* contains fresh randomness in each execution of e_A, for instance a nonce or an ephemeral public key.

The assumption that S_1 and S_2 do not contain **newChannel** guarantees that all channels are public. It is not a limitation in practice, because CryptoVerif does not support **newChannel** in protocol specifications; **newChannel** is used only in manual proofs. We require that the inputs and outputs use distinct channels with indices the current replication indices, to identify channels unambiguously. The assumption that there is no **defined** condition in Q_2 facilitates a renumbering of variables: the variables of Q_{2B} have indices $\widetilde{i}$ in S_2 but $\widetilde{i}'$ in $S_{composed}$. It is not a strong limitation since most usages of **find** with **defined** conditions can also be encoded using tables, and tables are not affected by the renumbering of variables.

Like Theorem 1, the first conclusion of Theorem 2 allows us to transfer security properties proved on S_2 to $S_{composed}$, this time by relying on Section 5.1. We cannot prove indistinguishability here, because the variables of Q_{2B} are renumbered as mentioned above: since these variables



where $m_2 = \mathsf{enc}(\mathsf{concat}(k_A, n), k_{lt}, r')$.

Figure 5: A simple key exchange protocol

may be public, the renumbering may affect the context as well. The second conclusion allows us to transfer security properties proved on S_1 to $S_{composed}$ by Lemma 1, provided they are proved with public variables including k'_A and k_B, since C'_4 uses k'_A and k_B.

In our TLS case study, we apply this theorem to perform most compositions: the handshakes with the record protocol, using a traffic secret as common key, as well as the handshake with pre-shared key with itself and the initial handshake with the handshake with pre-shared key, using the pre-shared key as common key. However, this theorem does not apply for the client early traffic secret *cets*, because of the possibility of replays. (Theorem 3 deals with this case.) The next example illustrates the theorem on a simpler case.

Example 3. *Let us suppose that there are no hash oracles and consider the following very simple key exchange protocol, also shown in Figure 5:*

$$\begin{aligned}
&S_1 = c_7(); \mathbf{new}\ k_{lt} : T; \overline{c_8}\langle\rangle;\\
&((!^{i_A \le n_A} c_9[i_A](n : T_{nonce}); \mathbf{new}\ k_A : T; \mathbf{new}\ r' : T_r; \mathbf{let}\ m_2 = \mathsf{enc}(\mathsf{concat}(k_A, n), k_{lt}, r')\ \mathbf{in}\\
&\quad \mathbf{event}\ e_A((n, m_2), k_A, i_A); \mathbf{let}\ k'_A = k_A\ \mathbf{in}\ \overline{c_A[i_A]}\langle m_2\rangle)\\
&\ |\\
&\ (!^{i_B \le n_B} c_{10}[i_B](); \mathbf{new}\ n : T_{nonce}; \overline{c_{11}[i_B]}\langle n\rangle;\\
&\quad c_{12}[i_B](m_2 : \mathit{bitstring}); \mathbf{let}\ \mathsf{i}_\perp(\mathsf{concat}(k_B, {=}n)) = \mathsf{dec}(m_2, k_{lt})\ \mathbf{in}\\
&\quad \mathbf{event}\ e_B((n, m_2), k_B); \overline{c_B[i_B]}\langle\rangle))
\end{aligned}$$

After an input on channel c_7, this process generates a long-term key k_{lt} shared between A and B, returns control to the adversary by outputting on channel c_8, and runs the participants A and B in parallel. The participant B (at the bottom) is run at most n_B times. It waits for an input on channel $c_{10}[i_B]$, generates a fresh nonce n and sends it to A on channel $c_{11}[i_B]$. If the session runs normally, the adversary forwards this nonce to channel $c_9[i_A]$ for some i_A, so that A receives it, generates a fresh key k_A, and computes the message m_2 that is the encryption of k_A and n under k_{lt}. (The function concat *is concatenation.) Then, A executes the event e_A to record that it accepts the key k_A, in a session of session identifier (n, m_2). (In this example, the function* sid *is the pair.) It stores k_A in k'_A and sends message m_2 on channel $c_A[i_A]$. If the session runs normally, the adversary forwards this message to channel $c_{12}[i_B]$, so that B receives it, decrypts it, and in case of success, executes event e_B to record that it terminates with key $k_B = k_A$, in a session of session identifier (n, m_2).*

Assuming that $(\mathsf{enc}, \mathsf{dec})$ *is an authenticated encryption scheme, CryptoVerif shows that S_1 preserves the secrecy of k'_A up to probability p and satisfies* (2) *and* (3) *with public variables k'_A, k_B up to probabilities p' and p'' respectively, which depend on the probability of breaking the encryption scheme.*

We compose S_1 with the system $S^b_{2!}$ of Example 2. The syntactic assumptions are easy to *check, and the composed system is*

$$\begin{aligned}
&S^b_{composed} = c_7(); \textbf{new } k_{lt} : T; \overline{c_8}\langle\rangle;\\
&\quad((!^{i_A \le n_A} c_9[i_A](n : T_{nonce}); \textbf{new } k_A : T; \textbf{new } r' : T_r; \textbf{let } m_2 = \mathsf{enc}(\mathsf{concat}(k_A, n), k_{lt}, r') \textbf{ in}\\
&\quad\quad \textbf{event } e_A((n, m_2), k_A, i_A); \overline{c'_A[i_A]}\langle m_2\rangle;\\
&\quad\quad c_4[i_A]((m_0 : T_m, m_1 : T_m)); \textbf{new } r : T_r; \textbf{event } \mathsf{send}((n, m_2), m_b); \overline{c_5[i_A]}\langle\mathsf{enc}(m_b, k_A, r)\rangle)\\
&\quad|\\
&\quad(!^{i_B \le n_B} c_{10}[i_B](); \textbf{new } n : T_{nonce}; \overline{c_{11}[i_B]}\langle n\rangle;\\
&\quad\quad c_{12}[i_B](m_2 : \mathit{bitstring}); \textbf{let } \mathsf{i}_\perp(\mathsf{concat}(k_B, {=}n)) = \mathsf{dec}(m_2, k_{lt}) \textbf{ in}\\
&\quad\quad \textbf{event } e_B((n, m_2), k_B); \overline{c'_B[i_B]}\langle\rangle;\\
&\quad\quad c_6[i_B](y : \mathit{bitstring}); \textbf{let } m = \mathsf{dec}(y, k_B) \textbf{ in event } \mathsf{recv}((n, m_2), m)))
\end{aligned}$$

The composed protocol runs the key exchange as before, then it sends m_b encrypted, as in S_2. However, in A, it executes event send *with session identifier (n, m_2) and encrypts with key k_A. In B, it executes event* recv *with session identifier (n, m_2) and decrypts with key k_B. These values are provided by the key exchange protocol. (The processes Q_{1A} and Q_{1B} are the process 0 that does nothing, so we simply omit them.)*

Let $S^{b,\circ}_{composed}$ be obtained from $S^b_{composed}$ by removing events e_A and e_B. Let $t_1 = t_2 = n_A t_{\mathsf{enc}} + n_B t_{\mathsf{dec}}$. By Theorem 2, item 1), for $b \in \{0,1\}$, there exists f such that $S^{b,\circ}_{composed} \overset{\sim\,\emptyset,\emptyset}{\rightarrow}_{f,p_3} S^b_{2!}$ where $p_3(C_3, t_D) = p(C'_3 + t_D) + p'(C'_3, t_D) + p''(C'_3, t_D)$, C'_3 runs in time at most $t_{C_3} + t_2$, $f(C_3)$ runs in time at most $t_{C_3} + t_1$, and their other parameters are the same as those of C_3. Since f does not depend on the details of S^b_2, f does not depend on b. Since $S^0_{2!} \approx_{p'_1} S^1_{2!}$, by Lemma 2, $S^{0,\circ}_{composed} \approx_{p_4} S^{1,\circ}_{composed}$ where $p_4(C, t_D) = 2p_3(C, t_D) + p'_1(f(C), t_D)$. Since $S^b_{2!}$ satisfies (1) up to probability p'_2, by Lemma 3, $S^{b,\circ}_{composed}$ satisfies (1) up to probability $p_5(C) = p_3(C, t_{corr}) + p'_2(f(C))$, and so does $S^b_{composed}$.

By Theorem 2, item 2), there exist evaluation contexts C'^b_4 such that $S^b_{composed} \approx_0 C'^b_4[S_1]$ and C'^b_4 is acceptable for S_1 with public variables k'_A, k_B, runs in time at most t_2, and does not alter the other parameters. Since S_1 satisfies (2) and (3) with public variables k'_A, k_B up to probabilities p' and p'' respectively, by Lemma 1, $C'^b_4[S_1]$ and $S^b_{composed}$ satisfy (2) and (3) up to probabilities $p'_1(C) = p'(C[C'^b_4])$ and $p''_1(C) = p''(C[C'^b_4])$ respectively. Therefore, we transferred security properties from S_1 and $S^b_{2!}$ to the composed system.

The properties required on S_1 are closely related to the security of a key exchange protocol as defined in CryptoVerif [11]. As in [11], we require secrecy of the key obtained by A and injective authentication of A to B (2). The security of a key exchange includes mutual authentication, which is not needed for the composition. The correspondence (3) does not appear in [11]. Combined with (2), it implies that sessions that share the same session identifier have the same key:

$$\textbf{event}(e_A(\mathit{sid}, k_A, \widetilde{i}_1)) \wedge \textbf{event}(e_B(\mathit{sid}, k_B)) \Longrightarrow k_A = k_B \tag{4}$$

a property included in the definition of a key exchange in CryptoVerif [11]. Indeed, if $e_A(\mathit{sid}, k_A, \widetilde{i}_1)$ and $e_B(\mathit{sid}, k_B)$ are executed, then $e_A(\mathit{sid}, k_B, \widetilde{i}_2)$ is also executed for some $\widetilde{i}_2$ by (2), which implies $\widetilde{i}_1 = \widetilde{i}_2$ by (3), so the two events $e_A(\mathit{sid}, k_A, \widetilde{i}_1)$ and $e_A(\mathit{sid}, k_B, \widetilde{i}_2)$ are actually the same event, so $k_A = k_B$. The converse is not true in general, because (2) and (4) put no constraints in case event e_B is not executed with the considered session identifier.

These properties are also closely related to the properties used in previous composition results [16, 27, 29]. These results require key secrecy, as well as partnering or match security, which provides guarantees similar to (2) and (3). In particular, [16, 29] require a public session matching algorithm, that is, the adversary knows which sessions are partnered. We also have this property: sessions are partnered when they have the same session identifier, and the session identifier is computed from public messages $\widetilde{msg}_A$ and $\widetilde{msg}_B$ by the function sid. This property is relaxed in [27]: they allow to use keys of early stages (which are virtually revealed) in the session matching. In TLS 1.3, the handshake is encrypted, and the session matching should be done on the plaintext, so the handshake keys are indeed needed for the session matching. In the model we consider, instead of encrypting the handshake, the handshake keys are given to the adversary, so that it can encrypt and decrypt messages. The session matching can then be done with public data.

However, the required properties still differ from [16, 27, 29] in their presentation. We make explicit the distinction between the two participants of the protocol, and (3) requires that A executes at most one session with a given session identifier. By (2), we obtain that B also executes at most one session with a given identifier. In contrast, [16, 27, 29] require that there are at most two sessions with the same identifier, without distinguishing A and B. The correspondence (2) guarantees that these two sessions have the same key, which is also required by [16, 27, 29].

Our definition of the key exchange protocol S_1 allows much flexibility. In contrast to [16, 27, 29], we do not assume that the key exchange protocol is a public-key protocol. In TLS 1.3, the handshake with pre-shared key indeed relies on a shared key, and may not need a long-term public key. We encode "corrupt" queries, used to corrupt long-term keys, for instance to model forward secrecy, inside the context C. That allows us to deal with protocols that satisfy forward secrecy or not, without explicit distinction, in contrast to what [16, 27, 29] do. That also allows us to support keys that are forward secret only from a certain stage, as well as temporary keys, used in several sessions but not leaked by "corrupt" queries because their lifetime is short, as in the multi-stage framework of [27, 29]. As in [27, 29], the key exchange may continue running after accepting a key: it may send messages M_A and M_B and execute Q_{1A} and Q_{1B}. We allow composition with keys that are established before the last key exchange message, provided they are not used in the key exchange protocol. However, we cannot perform test queries on a stage-i key and still use it in the key exchange protocol; while [27, 29] allow that, they do not allow composition for such keys. (In their vocabulary, these keys are not *final*.) As in [27], the communication partner does not need to be known at the start of the protocol. Finally, we support key exchange protocols that guarantee mutual authentication, unilateral authentication, or no authentication, as [27, 29]. This point may seem counter-intuitive, since (2) requires unilateral authentication. However, the security properties are obviously proved only when the partner is honest. Therefore, the system S_1 executes the events e_A, e_B and stores the key in k'_A only when the partner is honest. (That can be tested using **find** or tables when the partner is not authenticated.) Then, the correspondence (2) holds, and we can apply Theorem 2. When the partner is dishonest, we simply leak the key. Since no security property is desired in this case, we can trivially compose with any protocol that uses this key. This situation appears in TLS, when the client is not authenticated. In this case, the server considers that its partner is honest when the Diffie-Hellman share it receives has been sent by the honest client. This condition replaces client authentication and allows CryptoVerif to prove (2).

We give a proof sketch of Theorem 2 here. The full proof appears in Appendix A.9.

Proof sketch. Let

$$
\begin{aligned}
G_1 = C''_{\mathsf{h}}[C[&\textbf{event } e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i});\\
&\textbf{find } \widetilde{u}'' = \widetilde{i}''' \leq \widetilde{n} \textbf{ suchthat defined}(\widetilde{msg}_A[\widetilde{i}'''], k[\widetilde{i}''']) \wedge\\
&\quad \mathsf{sid}(\widetilde{msg}_A) = \mathsf{sid}(\widetilde{msg}_A[\widetilde{i}''']) \textbf{ then yield else}\\
&\textbf{new } k : T; \overline{c'_A[\widetilde{i}]}\langle M_A\rangle; (Q_{1A} \mid Q'_{2A}\{\mathsf{sid}(\widetilde{msg}_A)/x\}),
\end{aligned}
$$

$$\begin{array}{l}\textbf{event } e_B(\mathsf{sid}(\widetilde{msg}_B), k_B);\\ \quad \textbf{find } \widetilde{u} = \widetilde{i''} \leq \widetilde{n} \textbf{ suchthat defined}(\widetilde{msg}_A[\widetilde{i''}], k[\widetilde{i''}]) \wedge\\ \quad\quad \mathsf{sid}(\widetilde{msg}_A[\widetilde{i''}]) = \mathsf{sid}(\widetilde{msg}_B) \wedge \mathsf{fresh}(\widetilde{i''}, \widetilde{u}) \textbf{ then}\\ \quad \overline{c'_B[\widetilde{i'}]}\langle M_B\rangle; (Q_{1B} \mid Q'_{2B}\{k[\widetilde{u}]/k, \mathsf{sid}(\widetilde{msg}_B)/x\})]]\end{array}$$

where $\mathsf{fresh}(\widetilde{i''}, \widetilde{u}) = \textbf{find } \widetilde{u'} = \widetilde{i'''} \leq \widetilde{n'} \textbf{ suchthat defined}(\widetilde{u}[\widetilde{i'''}]) \wedge \widetilde{u}[\widetilde{i'''}] = \widetilde{i''} \textbf{ then } \mathsf{false} \textbf{ else } \mathsf{true}$. We have $\mathsf{fresh}(\widetilde{i''}, \widetilde{u})$ when $\widetilde{i''}$ was not used before, that is, it does not occur in the array $\widetilde{u}$. The game G_1 runs the key exchange protocol followed by the protocol that uses the key, much like $S_{composed}$. However, in the participant A (after event e_A), it does not run the protocol that uses the key when the same session identifier has already been seen in a previous session (**find** $\widetilde{u}''$), and it generates a fresh key k instead of using the key provided by the key exchange protocol (**new** $k : T$). In the participant B (after event e_B), it gets the key that has been generated in A with the same session identifier (**find** $\widetilde{u}$), and requires that the key of a given session of A is reused at most once by B (condition $\mathsf{fresh}(\widetilde{i''}, \widetilde{u})$).

Let $Q'_2 = \mathsf{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c'_1, T_{\mathsf{sid}}, Q_2)$, so that $S_2 = C'_{\mathsf{h}}[Q'_2]$. The first step of the proof consists in showing that $G_1 \overset{\sim}{\rightarrow}{}^{V_1, V_1 \cup \{\widetilde{u}\}}_{f',0} C''_{\mathsf{h}}[C_5[Q'_2]]$ for some evaluation context C_5. This is proved by establishing a correspondence between the traces of these two games. In this step, we renumber the variables of Q'_{2B}, replacing indices $\widetilde{a} \leq \widetilde{n}'$ in G_1 with $\widetilde{u}[\widetilde{a}] \leq \widetilde{n}$ in $C''_{\mathsf{h}}[C_5[Q'_2]]$. (The condition $\mathsf{fresh}(\widetilde{i''}, \widetilde{u})$ guarantees that $\widetilde{u}$ never takes twice the same value, hence the function from $\widetilde{a}$ to $\widetilde{u}[\widetilde{a}]$ is injective. We exclude **defined** conditions in Q_2 to facilitate this renumbering.) Since some of these variables may be public, this renumbering may also affect the context around G_1 and $C''_{\mathsf{h}}[C_5[Q'_2]]$. That is why these two games are generally not indistinguishable.

Let G_1° (resp. C_5°) be obtained from G_1 (resp. C_5) by removing all events of S_1. By Appendix A.6, Lemma 11, we have

$$G_1^\circ \overset{\sim}{\rightarrow}{}^{V_1, V_1 \cup \{\widetilde{u}\}}_{f',0} C''_{\mathsf{h}}[C_5^\circ[Q'_2]] \tag{5}$$

By Lemma 4, there exists an evaluation context C'_5 such that

$$C''_{\mathsf{h}}[C_5^\circ[Q'_2]] \approx_0^{V_1 \cup \{\widetilde{u}\}} C'_5[C'_{\mathsf{h}}[Q'_2]] = C'_5[S_2] \tag{6}$$

where the context C'_5 runs in time at most $t_{C_5^\circ} \leq t_1$, calls the l-th hash oracle in C'_{h} at most $n_{\mathsf{h},l,C_5^\circ} \leq n_{\mathsf{h},l,1}$ times, so $n'_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,1}$, and its other parameters are the same as those of C_5°.

The proof that $S^\circ_{composed} \overset{\sim}{\rightarrow}{}^{V_1, V_2}_{f,p_3} S_2$ then proceeds in two main steps:

1. First, we write the process G_1 above as an evaluation context around

$$\begin{array}{c} S'_1 = C_{\mathsf{h}}[C[\textbf{event } e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \textbf{new } k'_A : T; \overline{c''_A[\widetilde{i}]}\langle (k'_A, M_A)\rangle; Q_{1A},\\ \textbf{event } e_B(\mathsf{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\widetilde{i'}]}\langle M_B\rangle; Q_{1B}]]\end{array}$$

   Let S''_1 be obtained by replacing **new** $k'_A : T$ with **let** $k'_A = k_A$ **in** in S'_1. Let G_2 be obtained by replacing **new** $k : T$ with **let** $k = k_A$ **in** in G_1. Let G_2°, ${S'_1}^\circ$, and ${S''_1}^\circ$ be obtained from G_2, S'_1, and S''_1 respectively by removing all events of S_1. Since S_1 preserves the secrecy of k'_A with public variables V ($k_A, k'_A \notin V$), ${S'_1}^\circ$ is indistinguishable from ${S''_1}^\circ$ (Appendix A.3, Lemma 9), so G_1° is indistinguishable from G_2°.

2. Second, we write G_2 as an evaluation context around S_1. Since S_1 satisfies the correspondences (2) and (3), so does G_2, so we obtain that, up to a small probability, the **find** $\widetilde{u}''$ in G_2 fails and the **find** $\widetilde{u}$ succeeds with $k[\widetilde{u}] = k_B$. From that, we show that G_2 is indistinguishable from $S_{composed}$, so G_2° is indistinguishable from $S^\circ_{composed}$ (Appendix A.6, Lemma 11).

We conclude that $S^\circ_{composed} \stackrel{\sim}{\rightarrow}^{V_1,V_2}_{f,p_3} S_2$ with $f(C_3) = f'(C_3)[C'_5[\,]]$ by combining these results with (5) and (6).

The proof that $S_{composed} \approx_0^{V'} C'_4[S_1]$ is easier. Since k'_A and k_B are public variables, the context C'_4 can use them. Since $\widetilde{msg}_A$ and $\widetilde{msg}_B$ are sent on public channels, the context C'_4 also has access to them. Therefore, it can execute $Q'_{2A}\{k'_A/k, \mathsf{sid}(\widetilde{msg}_A)/x\}$ and $Q'_{2B}\{k_B/k, \mathsf{sid}(\widetilde{msg}_B)/x\}$ as the composed system does. □

5.5 Non-injective Variant

The next theorem is a variant of Theorem 2 with non-injective authentication. In this case, the process Q_{2B} may be executed several times for each key. Previous work [16, 27, 29] did not consider this case.

Theorem 3 (Non-injective variant). *The conclusion of Theorem 2 still holds with the following changes in the hypotheses:* $Q_2 = c_1(); \textbf{new}\ k : T; \overline{c_2}\langle\rangle; (Q_{2A} \mid !^{\widetilde{i}' \leq \widetilde{n}'} Q_{2B})$, $Q'_{2B} = \mathsf{AddIdxSid}(\emptyset \leq \emptyset, x : T_{\mathsf{sid}}, Q_{2B})$ *where the notation* $\emptyset$ *designates the empty sequence, and the correspondence*

$$\textbf{event}(e_B(sid, k)) \Longrightarrow \textbf{event}(e_A(sid, k, \widetilde{i})) \tag{7}$$

instead of (2).

In the theorem above, the system S_1 satisfies non-injective authentication: the correspondence (7) means that for each execution of $e_B(sid, k)$, there is an execution of $e_A(sid, k, \widetilde{i})$. However, event $e_B(sid, k)$ can be executed several times for each execution of $e_A(sid, k, \widetilde{i})$. To compensate for that, the process Q_{2B} in Q_2, inside the system S_2, is replicated: it is under the replication $!^{\widetilde{i}' \leq \widetilde{n}'}$, with the same indices as those above e_B, so that it can also be executed several times for each shared key k. In the construction of the composed system, in Q'_{2B}, we do not need to add replication indices to Q_{2B}, since Q_{2B} already contains the replication indices $\widetilde{i}' \leq \widetilde{n}'$, because Q_{2B} is under the replication $!^{\widetilde{i}' \leq \widetilde{n}'}$. Hence, the construction of Q'_{2B} from Q_{2B} just adds the session identifier x.

In our TLS case study, we use this theorem to compose the handshake with pre-shared key with the record protocol using the client early traffic secret *cets* as common key. This theorem is needed because, in case `ClientHello` messages are replayed, several sessions of the server may obtain the same client early traffic secret, so the handshake does not guarantee injective authentication.

6 Application to TLS 1.3

In this section, we sketch the application of our composition theorems in order to compose the protocol pieces of TLS 1.3 as outlined in Figure 1. More details are given in Appendix B.2. The composition theorems are generally easy to apply: their assumptions are either proved by CryptoVerif or syntactic and easy to verify, and the composed protocol is syntactically built from the two pieces that we compose. The TLS case study still presents two additional difficulties:

- We compose protocols recursively an arbitrary number of times, in case there are successive handshakes with pre-shared keys or key updates in the record protocol, so we perform proofs by induction.
- The secrecy of payload messages is expressed by the secrecy of a bit b in a process that sends message m_b encrypted. We translate that into an indistinguishability between the process that sends m_0 and the one that sends m_1 (as $S_2^0 \approx_{p_1} S_2^1$ in Example 1). Then we perform compositions on these two processes and combine the obtained results in order to prove secrecy of messages for composed processes.

The length of the composition proof is mostly due to the number of compositions that we perform between the various protocol pieces and the number of properties that we prove about these protocols.

In the composition, we first compose the record protocol with itself recursively by Theorem 1, using the secrecy of the updated traffic secret, to show that the security properties of the record protocol are preserved by key updates. We obtain a model of the record protocol that performs at most m key updates, for any m. We perform similar compositions for the 0-RTT variants. We put these protocols under replication by Lemmas 5 and 6, to model several sessions of the record protocol with independent traffic secrets.

Second, we compose the handshake with pre-shared key with the record protocol, using secret keys *cats* and *sats*, by Theorem 2. We also compose them with secret key *cets*, using Theorem 3 and the first 0-RTT variant of the record protocol, mentioned in Section 3, when the `ClientHello` message is unaltered, and using Theorem 5 (shown in Appendix A.12) and the second 0-RTT variant when the `ClientHello` message is altered. We also compose the obtained process with itself recursively, using the resumption secret *resumption_secret* as pre-shared key in the next handshake, by Theorem 2, and put it under replication by Lemmas 5 and 6. These compositions yield processes that perform at most l successive handshakes with pre-shared key and m key updates.

Third, we compose the initial handshake with the record protocol, using secret keys *cats* and *sats*, by Theorem 2. We also compose the initial handshake with the process that runs handshakes with pre-shared key, using the resumption secret *resumption_secret* as pre-shared key, by Theorem 2.

In all these compositions, CryptoVerif proves all secrecy and correspondence properties required by the theorems. The composed protocol inherits security properties from the components we compose. Therefore, these compositions allow us to infer security properties of the TLS protocol from properties of the handshakes and the record protocol. In particular, we obtain message secrecy, message forward secrecy (with respect to the compromise of long-term client and server keys), and injective message authentication for non-0-RTT application messages in both directions. For 0-RTT messages, since the handshake does not prevent replays for *cets*, we obtain non-injective authentication instead of the injective one. The correspondence properties of the handshakes are inherited by the composition and we also obtain secrecy of the exported master secrets *ems* provided by the various handshakes.

7 Conclusion

This paper presents several composition theorems, to compose a protocol that provides a key (e.g., a key exchange protocol) with a protocol that uses this key. These theorems rely on the computational model of cryptography. They are expressed in the framework of the tool CryptoVerif, so they are easily applicable when each protocol to compose is proved secure by

CryptoVerif. They provide great flexibility. In particular, they allow the composed protocols to share hash oracles, and they support non-injective as well as injective authentication.

We apply these theorems to TLS 1.3. This is an important case study, which illustrates well the power of our results. It allows us to prove security for any number of successive handshakes and key updates, a result that would be out of scope of CryptoVerif without composition, because this tool does not support loops. However, our theorems are of much more general interest, and we expect them to be applied to other protocols in the future. For instance, they apply as soon as a key exchange protocol provides a key to a cleanly separated transport protocol, a situation desirable in the design of many protocols.

Our results are specific to the CryptoVerif tool. We see no obstacle to recasting them in the framework of other tools that perform proofs in the computational model, such as EasyCrypt [5, 6]. However, although the general approach would be the same, a lot of our work would probably have to be redone to adapt the result to the language and formalism of each new tool. The assumptions of our theorems are either proved by CryptoVerif or syntactic and easy to verify. If desired, it would not be difficult to automate their verification and the application of the theorems in CryptoVerif. However, automating the application to TLS 1.3 would be more complicated, due to the additional difficulties mentioned at the beginning of Section 6. An interesting future work would also be to prove composition results with a key exchange protocol that already uses the key, for instance for key confirmation, in the line of [15].

Acknowledgements This work was partly supported by ANR TECAP (decision number ANR-17-CE39-0004-03) and H2020 NEXTLEAP.

References

[1] M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. *IEE Proceedings Information Security*, 153(1):27–39, Mar. 2006.

[2] M. Arapinis, V. Cheval, and S. Delaune. Verifying privacy-type properties in a modular way. In *CSF'12*, pages 95–109. IEEE, June 2012.

[3] M. Arapinis, V. Cheval, and S. Delaune. Composing security protocols: from confidentiality to privacy. In R. Focardi and A. Myers, editors, *POST'15*, volume 9036 of *LNCS*, pages 324–343. Springer, Apr. 2015.

[4] G. Barthe, J. M. Crespo, Y. Lakhnech, and B. Schmidt. Mind the gap: Modular machine-checked proofs of one-round key exchange protocols. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9057 of *LNCS*, pages 689–718. Springer, Apr. 2015.

[5] G. Barthe, F. Dupressoir, B. Grégoire, C. Kunz, B. Schmidt, and P.-Y. Strub. EasyCrypt: A tutorial. In A. Aldini, J. Lopez, and F. Martinelli, editors, *Foundations of Security Analysis and Design VII*, volume 8604 of *LNCS*, pages 146–166. Springer, 2014.

[6] G. Barthe, B. Grégoire, S. Heraud, and S. Z. Béguelin. Computer-aided security proofs for the working cryptographer. In P. Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *LNCS*, pages 71–90. Springer, Aug. 2011.

[7] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Eurocrypt'00*, volume 1807 of *LNCS*, pages 139–155. Springer, 2000.

[8] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *Eurocrypt'06*, volume 4004 of *LNCS*, pages 409–426. Springer, May 2006. Extended version available at `http://ia.cr/2004/331`.

[9] K. Bhargavan, B. Blanchet, and N. Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In *S&P'17*, pages 483–503. IEEE, May 2017.

[10] K. Bhargavan, B. Blanchet, and N. Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. Research Report RR-9040, Inria, May 2017. Available at `https://hal.inria.fr/hal-01528752`. CryptoVerif scripts available at `https://github.com/Inria-Prosecco/reftls/tree/master/cv`.

[11] B. Blanchet. Computationally sound mechanized proofs of correspondence assertions. In *CSF'07*, pages 97–111. IEEE, July 2007. Extended version available as ePrint Report 2007/128, `http://ia.cr/2007/128`.

[12] B. Blanchet. A computationally sound mechanized prover for security protocols. *IEEE Transactions on Dependable and Secure Computing*, 5(4):193–207, Oct.–Dec. 2008.

[13] B. Blanchet. Automatically verified mechanized proof of one-encryption key exchange. In *CSF'12*, pages 325–339. IEEE, June 2012.

[14] B. Blanchet. CryptoVerif: A computationally-sound security protocol verifier. Available at `http://cryptoverif.inria.fr/cryptoverif.pdf`, 2017.

[15] C. Brzuska, M. Fischlin, N. P. Smart, B. Warinschi, and S. C. Williams. Less is more: relaxed yet composable security notions for key exchange. *International Journal of Information Security*, 12(4):267–297, Aug. 2013.

[16] C. Brzuska, M. Fischlin, B. Warinschi, and S. Williams. Composability of Bellare-Rogaway key exchange protocol. In *CCS'11*, pages 51–62. ACM, 2011.

[17] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS'01*, pages 136–145. IEEE, Oct. 2001. An updated version is available at Cryptology ePrint Archive, `http://ia.cr/2000/067`.

[18] R. Canetti and J. Herzog. Universally composable symbolic analysis of mutual authentication and key exchange protocols. In S. Halevi and T. Rabin, editors, *TCC'06*, volume 3876 of *LNCS*, pages 380–403. Springer, Mar. 2006. Extended version available at `http://ia.cr/2004/334`.

[19] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *Eurocrypt'01*, volume 2045 of *LNCS*, pages 453–474. Springer, May 2001. Long version at `https://ia.cr/2001/040`.

[20] R. Canetti and T. Rabin. Universal composition with joint state. In D. Boneh, editor, *Crypto'03*, volume 2729 of *LNCS*, pages 265–281. Springer, Aug. 2003.

[21] Ş. Ciobâcă and V. Cortier. Protocol composition for arbitrary primitives. In *CSF'10*, pages 322–336. IEEE, July 2010.

[22] V. Cortier and S. Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1):1–36, Feb. 2009.

[23] A. Datta, A. Derek, J. C. Mitchell, and A. Roy. Protocol composition logic (PCL). *ENTCS*, 172:311–358, Apr. 2007.

[24] A. Datta, A. Derek, J. C. Mitchell, and B. Warinschi. Computationally sound compositional logic for key exchange protocols. In *CSFW'06*, pages 321–334. IEEE, July 2006.

[25] S. Delaune, S. Kremer, and O. Pereira. Simulation based security in the applied pi calculus. In R. Kannan and K. Narayan Kumar, editors, *FSTTCS'09*, volume 4 of *Leibniz International Proceedings in Informatics*, pages 169–180. Leibniz-Zentrum für Informatik, Dec. 2009.

[26] S. Delaune, S. Kremer, and M. D. Ryan. Composition of password-based protocols. In *CSF'08*, pages 239–251. IEEE, June 2008.

[27] B. Dowling, M. Fischlin, F. Günther, and D. Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In *CCS'15*, pages 1197–1210, 2015. Full version available at `https://ia.cr/2015/914`.

[28] B. Dowling, M. Fischlin, F. Günther, and D. Stebila. A cryptographic analysis of the TLS 1.3 draft-10 full and pre-shared key handshake protocol. Cryptology ePrint Archive, Report 2016/081, 2016. `https://ia.cr/2016/081`.

[29] M. Fischlin and F. Günther. Multi-stage key exchange and the case of Google's QUIC protocol. In *CCS'14*, pages 1193–1204, 2014. Full version available at `http://www.cryptoplexity.informatik.tu-darmstadt.de/media/crypt/publications_1/fischlin-guenther-ccs2014.pdf`.

[30] T. Groß and S. Mödersheim. Vertical protocol composition. In *CSF'11*, pages 235–250. IEEE, June 2011.

[31] J. D. Guttman and F. J. T. Fábrega. Protocol independence through disjoint encryption. In *CSFW'00*, pages 24–34. IEEE, July 2000.

[32] R. Küsters and M. Tuengerthal. Composition theorems without pre-established session identifiers. In *CCS'11*, pages 41–50. ACM, 2011.

[33] S. Mödersheim and L. Viganò. Secure pseudonymous channels. In M. Backes and P. Ning, editors, *ESORICS'09*, volume 5789 of *LNCS*, pages 337–354. Springer, Sept. 2009.

[34] S. Mödersheim and L. Viganò. Sufficient conditions for vertical composition of security protocols. In *AsiaCCS'14*, pages 435–446. ACM, June 2014.

[35] E. Rescorla. The Transport Layer Security (TLS) protocol version 1.3, draft-ietf-tls-tls13-28. `https://tools.ietf.org/html/draft-ietf-tls-tls13-28`, Mar. 2018.

[36] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, Nov. 2004. Available at `http://ia.cr/2004/332`.

[37] T. Y. C. Woo and S. S. Lam. A semantic model for authentication protocols. In *S&P'93*, pages 178–194. IEEE, May 1993.

Appendices

A Proofs

A.1 Characterization of Secrecy

Lemma 7 below gives a characterization of secrecy, which was used as a definition in versions of CryptoVerif that did not include **event_abort** [11,12]. We use it as an intermediate step in our proofs.

Lemma 7. *Let Q be a process that does not contain* **event_abort**. *Let x and V be such that $x \notin V$. Let*

$$
\begin{aligned}
R_x^0 &= !^{i_s \leq n_s}\, c_s[i_s](\widetilde{u} \leq \widetilde{n}); \textbf{if defined}(x[\widetilde{u}]) \textbf{ then } \overline{c_s[i_s]}\langle x[\widetilde{u}]\rangle \\
R_x^1 &= !^{i_s \leq n_s}\, c_s[i_s](\widetilde{u} \leq \widetilde{n}); \textbf{if defined}(x[\widetilde{u}]) \textbf{ then} \\
&\quad \textbf{find } u_{s1} = i_{s1} \leq n_s \textbf{ suchthat defined}(y[i_{s1}], \widetilde{u}[i_{s1}]) \wedge \widetilde{u}[i_{s1}] = \widetilde{u} \\
&\quad \textbf{then } \overline{c_s[i_s]}\langle y[u_{s1}]\rangle \\
&\quad \textbf{else new } y : T; \overline{c_s[i_s]}\langle y\rangle
\end{aligned}
$$

where the channel c_s and the variables $\widetilde{u}, u_{s1}, y$ do not occur in Q and the variable x has type T and is defined under replications $!^{\widetilde{i} \leq \widetilde{n}}$ in Q. Let Q° be obtained from Q by removing all events.

If the process Q preserves the secrecy of x with public variables V up to probability p, then $Q^\circ \mid R_x^0 \approx_{p'}^V Q^\circ \mid R_x^1$, where $p'(C, t_D) = p(C + t_D)$ and the context $C + t_D$ runs in time at most $t_C + t_D$ and its other parameters are the same as those of C.

Conversely, if $Q^\circ \mid R_x^0 \approx_{p'}^V Q^\circ \mid R_x^1$, then Q preserves the secrecy of x with public variables V up to probability p, where $p(C) = p'(C, t_D)$ and the distinguisher D is true when a certain event has been executed.

The processes R_x^0 and R_x^1 allow the adversary to query the variable x: if the adversary sends indices $\widetilde{u}$ on channel $c_s[i_s]$, and $x[\widetilde{u}]$ is defined, then the process R_x^0 replies with the value of $x[\widetilde{u}]$; instead, the process R_x^1 replies with a random value. The **find** in R_x^1 makes sure that, if the indices $\widetilde{u}$ have already been queried, the previous reply is sent; otherwise, a fresh random value y is chosen in the type T of x by **new** $y : T$, and sent as a reply. Lemma 7 says that x is secret with public variables V if and only if an adversary with access to variables V cannot distinguish between $Q^\circ \mid R_x^0$ and $Q^\circ \mid R_x^1$, that is, it cannot distinguish between the real values of x and independent random values.

Proof. Let us first prove that, if Q preserves the secrecy of x with public variables V up to probability p, then $Q^\circ \mid R_x^0 \approx_{p'}^V Q^\circ \mid R_x^1$. Let C be an evaluation context acceptable for $Q^\circ \mid R_x^0$ and $Q^\circ \mid R_x^1$ with public variables V. Let c_{s0} and c'_s be channels that C does not use. We define a context C' ($C + t_D$ in the statement of the lemma) that behaves like C except that:

- C' starts by outputting on channel c_{s0} and inputting on channel c_{s0}, then it starts running C.
- When C executes an event **event** $e(M_1, \dots, M_l)$, C' collects the executed event in its global state.
- When C terminates, C' recovers the sequence $\mathcal{E}$ of executed events from its global state, and sends $D(\mathcal{E}, 0)$ on channel c'_s.

Such a context C' exists because it can be encoded as a probabilistic Turing machine adversary, which can itself be encoded as a context in CryptoVerif [14, Section 2.8].

By definition of secrecy (Definition 2), when b is true, $C'[Q \mid R_x]$ behaves like $C[Q^\circ \mid R_x^0]$ and executes event S if and only if true is sent on channel c'_s, that is $D(\mathcal{E}, 0)$ is true. (It is important that Q never aborts, so that $C'[Q \mid R_x]$ can be programmed never to abort as well, and always sends some message on c'_s.) So

$$\Pr[C[Q^\circ \mid R_x^0] : D] = \Pr[C'[Q \mid R_x] : \mathsf{S}/b = \mathsf{true}]\,.$$

Similarly, when b is false, $C'[Q \mid R_x]$ behaves like $C[Q^\circ \mid R_x^1]$ and executes event $\overline{\mathsf{S}}$ if and only if true is sent on channel c'_s, so

$$\Pr[C[Q^\circ \mid R_x^1] : D] = \Pr[C'[Q \mid R_x] : \overline{\mathsf{S}}/b = \mathsf{false}]\,.$$

Therefore,

$$\begin{aligned}
&\Pr[C[Q^\circ \mid R_x^0] : D] - \Pr[C[Q^\circ \mid R_x^1] : D]\\
&\quad = \Pr[C'[Q \mid R_x] : \mathsf{S}/b = \mathsf{true}] - \Pr[C'[Q \mid R_x] : \overline{\mathsf{S}}/b = \mathsf{false}]\\
&\quad = \Pr[C'[Q \mid R_x] : \mathsf{S} \wedge b = \mathsf{true}]/\Pr[b = \mathsf{true}] - \Pr[C'[Q \mid R_x] : \overline{\mathsf{S}} \wedge b = \mathsf{false}]/\Pr[b = \mathsf{false}]\\
&\quad = 2.\Pr[C'[Q \mid R_x] : \mathsf{S} \wedge b = \mathsf{true}] - 2.\Pr[C'[Q \mid R_x] : \overline{\mathsf{S}} \wedge b = \mathsf{false}]\\
&\quad = \Pr[C'[Q \mid R_x] : \mathsf{S} \wedge b = \mathsf{true}] + 1/2 - \Pr[C'[Q \mid R_x] : \overline{\mathsf{S}} \wedge b = \mathsf{true}] -\\
&\qquad (\Pr[C'[Q \mid R_x] : \overline{\mathsf{S}} \wedge b = \mathsf{false}] + 1/2 - \Pr[C'[Q \mid R_x] : \mathsf{S} \wedge b = \mathsf{false}])
\end{aligned}$$

noting that $\Pr[C'[Q \mid R_x] : \overline{\mathsf{S}} \wedge b = \mathsf{true}] + \Pr[C'[Q \mid R_x] : \mathsf{S} \wedge b = \mathsf{true}] = \Pr[b = \mathsf{true}] = 1/2$ since S or $\overline{\mathsf{S}}$ is always executed in this particular game, and similarly for $b = \mathsf{false}$. Then

$$\begin{aligned}
&\Pr[C[Q^\circ \mid R_x^0] : D] - \Pr[C[Q^\circ \mid R_x^1] : D]\\
&\quad = \Pr[C'[Q \mid R_x] : \mathsf{S} \wedge b = \mathsf{true}] - \Pr[C'[Q \mid R_x] : \overline{\mathsf{S}} \wedge b = \mathsf{true}] -\\
&\qquad \Pr[C'[Q \mid R_x] : \overline{\mathsf{S}} \wedge b = \mathsf{false}] + \Pr[C'[Q \mid R_x] : \mathsf{S} \wedge b = \mathsf{false}]\\
&\quad = \Pr[C'[Q \mid R_x] : \mathsf{S}] - \Pr[C'[Q \mid R_x] : \overline{\mathsf{S}}]\\
&\quad \le p(C') = p'(C, t_D)
\end{aligned}$$

In case $\Pr[C[Q^\circ \mid R_x^0] : D] - \Pr[C[Q^\circ \mid R_x^1] : D] < 0$, we consider the distinguisher $\neg D$ that returns the negation of D and obtain

$$\begin{aligned}
\Pr[C[Q^\circ \mid R_x^1] : D] - \Pr[C[Q^\circ \mid R_x^0] : D] &= \Pr[C[Q^\circ \mid R_x^0] : \neg D] - \Pr[C[Q^\circ \mid R_x^1] : \neg D]\\
&\le p'(C, t_D)
\end{aligned}$$

so we conclude that

$$|\Pr[C[Q^\circ \mid R_x^0] : D] - \Pr[C[Q^\circ \mid R_x^1] : D]| \le p'(C, t_D)$$

hence $Q^\circ \mid R_x^0 \approx_{p'} Q^\circ \mid R_x^1$.

Conversely, let us prove that, if $Q^\circ \mid R_x^0 \approx_{p'}^{V} Q^\circ \mid R_x^1$, then Q preserves the secrecy of x with public variables V up to probability p, where $p(C) = p'(C, t_D)$. Let C be an evaluation context acceptable for $Q \mid R_x$ with public variables V that does not contain S nor $\overline{\mathsf{S}}$. We define a context C' that behaves like C except that:

- When C sends a message on channel c_{s0}, C' immediately replies on channel c_{s0}.

- When C sends a bit b' on channel c'_s, if the bit b' is true, then C' executes **event_abort** e, else C' executes **event_abort** e'.
- When C terminates (without having sent a bit on c'_s), C' chooses a random bit b'; if b' is true, then C' executes **event_abort** e, else C' executes **event_abort** e'.

Let D be the distinguisher such that $D(\mathcal{E}, a)$ is true if and only if $\mathcal{E}$ contains event e.

When b is true, $C'[Q^\circ \mid R^0_x]$ behaves like $C[Q \mid R_x]$ and executes event e if and only if $b' = \mathsf{true}$ is sent on channel c'_s (that is, $b' = b$, that is, event S is executed) or $C[Q \mid R_x]$ terminates without having sent a bit on c'_s and the random bit b' is true. So

$$\Pr[C'[Q^\circ \mid R^0_x] : D] = \Pr[C[Q \mid R_x] : \mathsf{S}/b = \mathsf{true}] + 1/2 \Pr[C[Q \mid R_x] : \text{term.}/b = \mathsf{true}]\,.$$

(We write "term." for "terminates without having sent a bit on c'_s".) Similarly, when b is false, $C'[Q^\circ \mid R^1_x]$ behaves like $C[Q \mid R_x]$ and executes event e if and only if $b' = \mathsf{true}$ is sent on channel c'_s (that is, $b' \neq b$, that is, event $\overline{\mathsf{S}}$ is executed) or $C[Q \mid R_x]$ terminates without having sent a bit on c'_s and the random bit b' is true, so

$$\Pr[C'[Q^\circ \mid R^1_x] : D] = \Pr[C[Q \mid R_x] : \overline{\mathsf{S}}/b = \mathsf{false}] + 1/2 \Pr[C[Q \mid R_x] : \text{term.}/b = \mathsf{false}]\,.$$

Therefore,

$$\begin{aligned}
&\Pr[C'[Q^\circ \mid R^0_x] : D] - \Pr[C'[Q^\circ \mid R^1_x] : D] \\
&\quad = \Pr[C[Q \mid R_x] : \mathsf{S}/b = \mathsf{true}] - \Pr[C[Q \mid R_x] : \overline{\mathsf{S}}/b = \mathsf{false}] + \\
&\qquad 1/2 \Pr[C[Q \mid R_x] : \text{term.}/b = \mathsf{true}] - 1/2 \Pr[C[Q \mid R_x] : \text{term.}/b = \mathsf{false}] \\
&\quad = \Pr[C[Q \mid R_x] : \mathsf{S} \wedge b = \mathsf{true}]/\Pr[b = \mathsf{true}] - \Pr[C[Q \mid R_x] : \overline{\mathsf{S}} \wedge b = \mathsf{false}]/\Pr[b = \mathsf{false}] + \\
&\qquad 1/2 \Pr[C[Q \mid R_x] : \text{term.} \wedge b = \mathsf{true}]/\Pr[b = \mathsf{true}] - \\
&\qquad 1/2 \Pr[C[Q \mid R_x] : \text{term.} \wedge b = \mathsf{false}]/\Pr[b = \mathsf{false}] \\
&\quad = 2.\Pr[C[Q \mid R_x] : \mathsf{S} \wedge b = \mathsf{true}] - 2.\Pr[C[Q \mid R_x] : \overline{\mathsf{S}} \wedge b = \mathsf{false}] + \\
&\qquad \Pr[C[Q \mid R_x] : \text{term.} \wedge b = \mathsf{true}] - \Pr[C[Q \mid R_x] : \text{term.} \wedge b = \mathsf{false}] \\
&\quad = \Pr[C[Q \mid R_x] : \mathsf{S} \wedge b = \mathsf{true}] + 1/2 - \Pr[C[Q \mid R_x] : \overline{\mathsf{S}} \wedge b = \mathsf{true}] - \\
&\qquad (\Pr[C[Q \mid R_x] : \overline{\mathsf{S}} \wedge b = \mathsf{false}] + 1/2 - \Pr[C[Q \mid R_x] : \mathsf{S} \wedge b = \mathsf{false}])
\end{aligned}$$

noting that $\Pr[C[Q \mid R_x] : \overline{\mathsf{S}} \wedge b = \mathsf{true}] + \Pr[C[Q \mid R_x] : \mathsf{S} \wedge b = \mathsf{true}] + \Pr[C[Q \mid R_x] : \text{term.} \wedge b = \mathsf{true}] = \Pr[b = \mathsf{true}] = 1/2$ and similarly for $b = \mathsf{false}$. So

$$\Pr[C'[Q^\circ \mid R^0_x] : D] - \Pr[C'[Q^\circ \mid R^1_x] : D] = \Pr[C[Q \mid R_x] : \mathsf{S}] - \Pr[C[Q \mid R_x] : \overline{\mathsf{S}}]$$

Hence

$$\begin{aligned}
\Pr[C[Q \mid R_x] : \mathsf{S}] - \Pr[C[Q \mid R_x] : \overline{\mathsf{S}}] &= \Pr[C'[Q^\circ \mid R^0_x] : D] - \Pr[C'[Q^\circ \mid R^1_x] : D] \\
&\leq p'(C', t_D) = p(C')
\end{aligned}$$

All parameters (runtime, replication bounds, lengths of bitstrings) are the same for C' and for C (up to very small runtimes) so we obtain $\Pr[C[Q \mid R_x] : \mathsf{S}] - \Pr[C[Q \mid R_x] : \overline{\mathsf{S}}] \leq p(C)$. $\square$

A.2 Eliminating Private Communications

In this section, we prove indistinguishability results by eliminating communications on private channels. Let us consider the following simple example. In the next lemma, we write $P\{M/x\}$ for the process obtained from P by substituting M for x. Similarly, we will write $Q_0\{\mathsf{fr}_c/c, c \in S\}$ for the process obtained from Q_0 by renaming the channel c into fr_c for each $c \in S$.

Lemma 8. *Let C be any context with replications $!^{\widetilde{i}\leq\widetilde{n}}$ above the hole. Let P be a process, without* **defined** *conditions on the variable x. We have*

$$\textbf{newChannel}\ c; (C[\overline{c[\widetilde{i}]}\langle M\rangle] \mid !^{\widetilde{i}\leq\widetilde{n}} c[\widetilde{i}](x:T); P) \approx_0^V C[P\{M/x\}]$$

where the channel c does not occur elsewhere and V does not contain x.

The process on the left-hand side sends a message M on the private channel $c[\widetilde{i}]$, and upon reception, the message is stored in x and P is executed. In the right-hand side, we shortcut the private communication and directly execute P with M substituted for x.

Proof sketch. The proof consists in relating the traces of the two processes in the presence of an evaluation context. It is easy to see that traces of the same probability match in the two processes, since the output on channel $c[\widetilde{a}]$ must be received by the input on $c[\widetilde{a}]$. After that communication, the process P is then executed with the value of M for x. (Terms are deterministic, so evaluating M several times or none does not change the behavior of the processes.) $\square$

We could prove other indistinguishability results obtained by eliminating private communications, using the same proof technique. Proving a general lemma would yield complex notations, so we prefer just referring to this proof technique when we need it in the following composition results.

A.3 Consequence of Secrecy

The next lemma shows that, when a key k is secret, we can replace k with a fresh random key. The adversary cannot distinguish the real definition of k from the random one. In this lemma, we need to remove events, because events can leak information on k even when k is secret. That allows us to apply the previous characterization of secrecy (Lemma 7).

Lemma 9. *Let C be any context with replications $!^{\widetilde{i}\leq\widetilde{n}}$ above the hole and without* **event_abort**. *Let Q be a process without* **event_abort**. *Let C° and Q° be obtained from C and Q respectively by removing all events. Let M be a term of type T. Suppose that k does not occur in C, M, M', and Q. Let c_k, c'_k be channels that do not occur in C and Q. If $C[\textbf{let}\ k = M\ \textbf{in}\ \overline{c_k[\widetilde{i}]}\langle M'\rangle; Q]$ preserves the secrecy of k with public variables V ($k \notin V$) up to probability p, then*

$$C^\circ[\textbf{let}\ k = M\ \textbf{in}\ \overline{c'_k[\widetilde{i}]}\langle(k, M')\rangle; Q^\circ] \approx_{p'}^V C^\circ[\textbf{new}\ k:T; \overline{c'_k[\widetilde{i}]}\langle(k, M')\rangle; Q^\circ]$$

where $p'(C', t_D) = p(C' + t_D)$.

Proof. Let $G = C^\circ[\textbf{let}\ k = M\ \textbf{in}\ \overline{c_k[\widetilde{i}]}\langle M'\rangle; Q^\circ]$. Let c_s be a channel that does not occur in C and Q, k' be a variable that does not occur in C and Q. By Lemma 7, since G does not contain events, we have $G \mid R_x^0 \approx_{p'}^V G \mid R_x^1$. Suppose that M' has type T'. Let

$$\begin{aligned} C' = {} & \textbf{newChannel}\ c_k, c_s; \\ & (!^{\widetilde{i}\leq\widetilde{n}} c_k[\widetilde{i}](x:T'); \overline{c_s[\mathsf{encode}(\widetilde{i})]}\langle\widetilde{i}\rangle; c_s[\mathsf{encode}(\widetilde{i})](k':T); \overline{c'_k[\widetilde{i}]}\langle(k', x)\rangle \mid [\,]) \end{aligned}$$

where $\widetilde{n} = n_1, \dots, n_m$ and $\mathsf{encode}(\widetilde{i})$ encodes the tuple $\widetilde{i} \in [1, n_1] \times \cdots \times [1, n_m]$ as a element of $[1, n_1 \times \cdots \times n_m]$.

We have

$$C'[G \mid R_x^0] \approx_0^V C^\circ[\textbf{let}\ k = M\ \textbf{in}\ \overline{c'_k[\widetilde{i}]}\langle(k, M')\rangle; Q^\circ]$$

by eliminating communications on c_k, c_s (Appendix A.2) and similarly

$$C'[G \mid R_x^1] \approx_0^V C^\circ[\mathbf{new}\ k:T; \overline{c'_k[\widetilde{i}]}\langle(k,M')\rangle; Q^\circ]$$

since the variable $k[\widetilde{i}]$ is always defined when a message is sent on channel $c_k[\widetilde{i}]$, so when R_x^1 tests that definition, and furthermore, the indices $\widetilde{i}$ are sent at most once on channel $c_s[\dots]$, so R_x^1 always replies with a fresh random element of type T. Moreover, since $G \mid R_x^0 \approx_{p'}^V G \mid R_x^1$, we have

$$C'[G \mid R_x^0] \approx_{p'}^V C'[G \mid R_x^1]$$

(we ignore the small runtime of C'), so we obtain the desired result by transitivity of $\approx$. □

A.4 Proof for Section 4

Proof of Theorem 1. Let C° and Q_1° be obtained from C and Q_1 respectively, by removing all events. Let $C' = \mathbf{newChannel}\ c_2, c_3; (C^\circ[\overline{c_2}\langle\rangle; c_3(); \overline{c'_1}\langle\rangle; Q_1^\circ] \mid [\,])$. Let c''_1 be a fresh channel. We have

$$\begin{aligned}
C'[S_2] &\approx_0^{V_1} C^\circ[\mathbf{new}\ k:T; \overline{c'_1}\langle\rangle; (Q_1^\circ \mid Q_2)] \\
&\qquad \text{by eliminating communications on channels } c_2 \text{ and } c_3 \text{ (Appendix A.2)} \\
&\approx_0^{V_1} \mathbf{newChannel}\ c''_1; (C^\circ[\mathbf{new}\ k:T; \overline{c''_1}\langle k\rangle; Q_1^\circ] \mid c''_1(k:T); \overline{c'_1}\langle\rangle; Q_2) \\
&\qquad \text{by eliminating communications on channel } c''_1 \text{ (Appendix A.2)} \\
&\approx_{p'}^{V_1} \mathbf{newChannel}\ c''_1; (C^\circ[\mathbf{let}\ k = M\ \mathbf{in}\ \overline{c''_1}\langle k\rangle; Q_1^\circ] \mid c''_1(k:T); \overline{c'_1}\langle\rangle; Q_2) \\
&\qquad \text{by Lemma 9 (We ignore the runtime of communication on } c''_1\text{.)} \\
&\approx_0^{V_1} S^\circ_{composed} \qquad \text{by eliminating communications on channel } c''_1 \text{ (Appendix A.2)}
\end{aligned}$$

Let k' be a fresh variable not in V'. Let

$$C'' = \mathbf{newChannel}\ c_1; (c_1(); \mathbf{if\ defined}(k)\ \mathbf{then\ let}\ k' = k\ \mathbf{in}\ \overline{c'_1}\langle\rangle; Q_2\{k'/k\} \mid [\,])$$

We have

$$\begin{aligned}
C''[S_1] &\approx_0^{V'} C[\mathbf{let}\ k = M\ \mathbf{in\ if\ defined}(k)\ \mathbf{then\ let}\ k' = k\ \mathbf{in}\ \overline{c'_1}\langle\rangle; (Q_1 \mid Q_2\{k'/k\})] \\
&\qquad \text{by eliminating communications on channel } c_1 \text{ (Appendix A.2)} \\
&\approx_0^{V'} C[\mathbf{let}\ k = M\ \mathbf{in}\ \overline{c'_1}\langle\rangle; (Q_1 \mid Q_2)] = S_{composed}
\end{aligned}$$

since the condition **defined**(k) is always true and $k' = k$. □

A.5 Proofs for Section 5.1

Proof of Lemma 2. Let C be any evaluation context acceptable for Q_1 and Q_2 with public variables V, and D be any distinguisher that runs in time at most t_D. Let $C' = f(C)$. The context C is an evaluation context acceptable for Q'_1 and Q'_2 with public variables V'. We have

$$\begin{aligned}
&|\Pr[C[Q_1]:D] - \Pr[C[Q_2]:D]| \\
&\quad \le |\Pr[C[Q_1]:D] - \Pr[C'[Q'_1]:D]| + |\Pr[C'[Q'_1]:D] - \Pr[C'[Q'_2]:D]| + \\
&\qquad |\Pr[C'[Q'_2]:D] - \Pr[C[Q_2]:D]| \\
&\quad \le p_1(C, t_D) + p'(f(C), t_D) + p_2(C, t_D) \\
&\quad \le p''(C, t_D)
\end{aligned}$$

□

Proof of Lemma 3. Let C be any evaluation context acceptable for Q with public variables V that does not contain events used in *corr*. Let $C' = f(C)$. The context C' is any evaluation context acceptable for Q' with public variables V' that does not contain events used in *corr*. We have

$$\begin{aligned}\Pr[C[Q] : \neg \mathit{corr}] &\leq |\Pr[C[Q] : \neg \mathit{corr}] - \Pr[C'[Q'] : \neg \mathit{corr}]| + \Pr[C'[Q'] : \neg \mathit{corr}]\\ &\leq p(C, t_{\mathit{corr}}) + p'(f(C)) = p''(C)\end{aligned}$$

□

Lemma 10. *If $Q \overset{V,V'}{\underset{f,p}{\tilde{\to}}} Q'$, $x \in V$, $x' \in V'$, $V \cap \mathsf{var}(R_x) = \{x\}$, and for every C_0 evaluation context acceptable for $Q \mid R_x$ with public variables $V \setminus \{x\}$, there exist C'_0 and C''_0 such that $f(C_0[[] \mid R_x]) = C'_0[C''_0[] \mid R_{x'}]$, $(\mathsf{var}(C'_0) \cup \mathsf{var}(C''_0)) \cap \mathsf{var}(R_{x'}) = \emptyset$, $\mathsf{vardef}(C'_0) \cap \mathsf{var}(C''_0) = \emptyset$, and C'_0 and C''_0 do not use any common table, then f is secrecy-preserving for $x' \mapsto (x, f_{\mathrm{sec}})$ with $f_{\mathrm{sec}}(C_0) = C'_0[C''_0]$.*

The set $\mathsf{vardef}(C'_0)$ contains the variables defined in C'_0.

Proof. Suppose $Q \overset{V,V'}{\underset{f,p}{\tilde{\to}}} Q'$, $x \in V$, $x' \in V'$, and Q' preserves the secrecy of x' with public variables $V' \setminus \{x'\}$ up to probability p'. Let C_0 be any evaluation context acceptable for $Q \mid R_x$ with public variables $V \setminus \{x\}$. Let C'_0 and C''_0 be such that $f(C_0[[] \mid R_x]) = C'_0[C''_0[] \mid R_{x'}]$, $(\mathsf{var}(C'_0) \cup \mathsf{var}(C''_0)) \cap \mathsf{var}(R_{x'}) = \emptyset$, $\mathsf{vardef}(C'_0) \cap \mathsf{var}(C''_0) = \emptyset$, and C'_0 and C''_0 do not use any common table. The context $C'_0[C''_0[] \mid R_{x'}]$ is an evaluation context acceptable for Q' with public variables V'. Since $\mathsf{var}(C''_0) \cap \mathsf{var}(R_{x'}) = \emptyset$, C''_0 does not use x', so C''_0 is an evaluation context acceptable for Q' with public variables $V' \setminus \{x'\}$, so by Lemma 1, $C''_0[Q']$ preserves the secrecy of x' with public variables $(V' \cup \mathsf{var}(C''_0)) \setminus \{x'\}$ up to probability p_1 such that $p_1(C') = p'(C'[C''_0])$. Since $\mathsf{var}(C'_0) \cap \mathsf{var}(R_{x'}) = \emptyset$, C'_0 does not use x', so C'_0 is an evaluation context acceptable for $C''_0[Q'] \mid R_{x'}$ with public variables $(V' \cup \mathsf{var}(C''_0)) \setminus \{x'\}$. To see that, please see the definition of acceptable evaluation contexts [14, Definition 4] and note that we have $\mathsf{var}(C'_0) \cap \mathsf{var}(C''_0[Q'] \mid R_{x'}) \subseteq (\mathsf{var}(C'_0) \cap \mathsf{var}(C''_0)) \cup (\mathsf{var}(C'_0) \cap \mathsf{var}(Q')) \subseteq \mathsf{var}(C''_0) \cup V'$, $\mathsf{vardef}(C'_0) \cap (\mathsf{var}(C''_0) \cup V') = (\mathsf{vardef}(C'_0) \cap \mathsf{var}(C''_0)) \cup (\mathsf{vardef}(C'_0) \cap V') = \emptyset$, and C''_0 and $C''_0[Q'] \mid R_{x'}$ do not use any common table. We have

$$\begin{aligned}\Pr[C_0[Q \mid R_x] : \mathsf{S}] - \Pr[C_0[Q \mid R_x] : \overline{\mathsf{S}}] \leq\ & |\Pr[C_0[Q \mid R_x] : \mathsf{S}] - \Pr[C'_0[C''_0[Q'] \mid R_{x'}] : \mathsf{S}]| +\\ & \Pr[C'_0[C''_0[Q'] \mid R_{x'}] : \mathsf{S}] - \Pr[C'_0[C''_0[Q'] \mid R_{x'}] : \overline{\mathsf{S}}] +\\ & |\Pr[C'_0[C''_0[Q'] \mid R_{x'}] : \overline{\mathsf{S}}] - \Pr[C_0[Q \mid R_x] : \overline{\mathsf{S}}]|\\ \leq\ & p(C_0[[] \mid R_x], t_{\mathsf{S}}) + p_1(C'_0) + p(C_0[[] \mid R_x], t_{\overline{\mathsf{S}}})\\ \leq\ & 2p(C_0[[] \mid R_x], t_{\mathsf{S}}) + p'(f_{\mathrm{sec}}(C_0))\end{aligned}$$

since $t_{\mathsf{S}} = t_{\overline{\mathsf{S}}}$ and $p_1(C'_0) = p'(C'_0[C''_0]) = p'(f_{\mathrm{sec}}(C_0))$. We conclude that Q preserves the secrecy of x with public variables $V \setminus \{x\}$ up to probability p'', where $p''(C_0) = 2p(C_0[[] \mid R_x], t_{\mathsf{S}}) + p'(f_{\mathrm{sec}}(C_0))$. Therefore, f is secrecy-preserving for $x' \mapsto (x, f_{\mathrm{sec}})$. □

A.6 Removing Events

As in the first conclusion of Theorem 1, we sometimes need to remove events. The next lemma shows that indistinguishability and the transfer relation of Section 5.1 are preserved by removing events.

Lemma 11. *Let Q_1° and Q_2° be obtained from Q_1 and Q_2 respectively by removing events $e_1, \dots, e_n$. If $Q_1 \approx_p^V Q_2$, then $Q_1^\circ \approx_p^V Q_2^\circ$. If $Q_1 \stackrel{\sim}{\rightarrow}_{f,p}^{V,V'} Q_2$ and f commutes with renamings of events, then $Q_1^\circ \stackrel{\sim}{\rightarrow}_{f,p}^{V,V'} Q_2^\circ$.*

Proof. Let $r(\mathcal{E})$ be the sequence of events obtained by removing events $e_1, \dots, e_n$ from $\mathcal{E}$. Let α be a renaming of $e_1, \dots, e_n$ to fresh event names.

Let C be an evaluation context acceptable for Q_1° and Q_2° with public variables V and D be a distinguisher. The context $\alpha(C)$ is also acceptable for Q_1°, Q_2°, Q_1, and Q_2 with public variables V. We have

$$\begin{aligned}
|\Pr[C[Q_1^\circ] : D] - \Pr[C[Q_2^\circ] : D]| &= |\Pr[\alpha(C)[Q_1^\circ] : D \circ \alpha^{-1}] - \Pr[\alpha(C)[Q_2^\circ] : D \circ \alpha^{-1}]| \\
&= |\Pr[\alpha(C)[Q_1] : D \circ \alpha^{-1} \circ r] - \Pr[\alpha(C)[Q_2] : D \circ \alpha^{-1} \circ r]| \\
&\le p(C, t_D)
\end{aligned}$$

neglecting the runtime of α^{-1} and r and noticing that the renaming of events does not change the parameters of C, so $Q_1^\circ \approx_p^V Q_2^\circ$.

Let C be an evaluation context acceptable for Q_1° with public variables V and D be a distinguisher. The context $\alpha(C)$ is also acceptable for Q_1° and for Q_1 with public variables V, so letting $C' = f(C)$, the context $\alpha(C') = \alpha(f(C)) = f(\alpha(C))$ is acceptable for Q_2 and so for Q_2° with public variables V', so C' is also acceptable for Q_2° with public variables V'. We have

$$\begin{aligned}
|\Pr[C[Q_1^\circ] : D] - \Pr[C'[Q_2^\circ] : D]| &= |\Pr[\alpha(C)[Q_1^\circ] : D \circ \alpha^{-1}] - \Pr[\alpha(C')[Q_2^\circ] : D \circ \alpha^{-1}]| \\
&= |\Pr[\alpha(C)[Q_1] : D \circ \alpha^{-1} \circ r] - \Pr[\alpha(C')[Q_2] : D \circ \alpha^{-1} \circ r]| \\
&\le p(C, t_D)
\end{aligned}$$

so $Q_1^\circ \stackrel{\sim}{\rightarrow}_{f,p}^{V,V'} Q_2^\circ$. □

A.7 Proof for Section 5.2

of Lemma 4. The lemma holds trivially when there is no random oracle: taking $C' = C$, we have $C_{\mathsf{h}}[C[Q]] \approx_0^V C'[C'_{\mathsf{h}}[Q]]$ because $C_{\mathsf{h}}[C[Q]] = C[Q] = C'[Q] = C'[C'_{\mathsf{h}}[Q]]$. Let us now assume that there is at least one random oracle.

Suppose that $C = \mathbf{newChannel}\ \widetilde{c}; ([\,] \mid Q_1)$, with $(\{c_{\mathsf{h}1}, c_{\mathsf{h}2}, c'_{\mathsf{h}1}, c'_{\mathsf{h}2}\} \cup \{c_{\mathsf{h}3,l}, c_{\mathsf{h}4,l}, c'_{\mathsf{h}3,l}, c'_{\mathsf{h}4,l} \mid l \le L\}) \cap \{\widetilde{c}\} = \emptyset$. (We can generalize to any evaluation context by applying the result several times and by commuting parallel compositions if needed.)

Let Q'_1 be obtained from Q_1 by introducing assignments to fresh variables x_j ($j \ge 1$) so that all occurrences of $\mathsf{h}_l(hk_{\mathsf{h},l}, M_j)$ are in processes **let** $x_j = \mathsf{h}_l(hk_{\mathsf{h},l}, M_j)$ **in**, and replacing these processes with

$$\overline{c'_{\mathsf{h}3,l}[f_{l,j}(\widetilde{i})]}\langle M_j \rangle; c'_{\mathsf{h}4,l}[f_{l,j}(\widetilde{i})](x_j : T'_{\mathsf{h},l})$$

where $\widetilde{i}$ are the replication indices above **let** $x_j = \mathsf{h}_l(hk_{\mathsf{h},l}, M_j)$ **in** and the functions $f_{l,j}$ ($j \ge 1$) and the functions $f_{l,0}$ used below are chosen such that, for each $l \le L$, the function $(j, \widetilde{i}) \mapsto f_{l,j}(\widetilde{i})$ is injective.

Let $y_{\mathsf{h},l}$ and $y'_{\mathsf{h},l}$ be fresh variables. Let

$$\begin{aligned}
C' = {} & \mathbf{newChannel}\ c'_{\mathsf{h}1}, c'_{\mathsf{h}2}, c'_{\mathsf{h}3,l}, c'_{\mathsf{h}4,l}, \widetilde{c}; \\
& ([\,] \mid c_{\mathsf{h}1}(); \overline{c'_{\mathsf{h}1}}\langle\rangle; c'_{\mathsf{h}2}(); \overline{c_{\mathsf{h}2}}\langle\rangle; Q'_1 \mid \\
& \textstyle\prod_{l=1}^{L} !^{i_{\mathsf{h},l} \le n_{\mathsf{h},l}} c_{\mathsf{h}3,l}[i_{\mathsf{h},l}](y_{\mathsf{h},l} : T_{\mathsf{h},l}); \overline{c'_{\mathsf{h}3,l}[f_0(i_{\mathsf{h},l})]}\langle y_{\mathsf{h},l}\rangle; c'_{\mathsf{h}4,l}[f_0(i_{\mathsf{h},l})](y'_{\mathsf{h},l} : T'_{\mathsf{h},l}); \overline{c_{\mathsf{h}4,l}[i_{\mathsf{h},l}]}\langle y'_{\mathsf{h},l}\rangle)
\end{aligned}$$

We have

$$
\begin{aligned}
&C'[C'_{\mathsf{h}}[Q]]\\
&\quad\approx_0^V \textbf{newChannel } c'_{\mathsf{h}3,l}, c'_{\mathsf{h}4,l}, \widetilde{c};\\
&\qquad(c_{\mathsf{h}1}(); \textbf{new } hk_{\mathsf{h},1} : T_{hk_{\mathsf{h},1}}; \dots \textbf{new } hk_{\mathsf{h},L} : T_{hk_{\mathsf{h},L}}; \overline{c_{\mathsf{h}2}}\langle\rangle; (Q'_1 \mid Q \mid Q'_{\mathsf{h}}) \mid\\
&\qquad\prod_{l=1}^{L} !^{i_{\mathsf{h},l} \leq n_{\mathsf{h},l}} c_{\mathsf{h}3,l}[i_{\mathsf{h},l}](y_{\mathsf{h},l} : T_{\mathsf{h},l}); \overline{c'_{\mathsf{h}3,l}[f_0(i_{\mathsf{h},l})]}\langle y_{\mathsf{h},l}\rangle;\\
&\qquad\qquad c'_{\mathsf{h}4,l}[f_0(i_{\mathsf{h},l})](y'_{\mathsf{h},l} : T'_{\mathsf{h},l}); \overline{c_{\mathsf{h}4,l}[i_{\mathsf{h},l}]}\langle y'_{\mathsf{h},l}\rangle)\\
&\qquad\qquad\text{by eliminating communications on } c'_{\mathsf{h}1} \text{ and } c'_{\mathsf{h}2} \text{ (Appendix A.2)}\\
&\quad\approx_0^V \textbf{newChannel } \widetilde{c};\\
&\qquad(c_{\mathsf{h}1}(); \textbf{new } hk_{\mathsf{h},1} : T_{hk_{\mathsf{h},1}}; \dots \textbf{new } hk_{\mathsf{h},L} : T_{hk_{\mathsf{h},L}}; \overline{c_{\mathsf{h}2}}\langle\rangle; (Q_1 \mid Q \mid Q_{\mathsf{h}}))\\
&\qquad\text{by eliminating communications on } c'_{\mathsf{h}3,l} \text{ and } c'_{\mathsf{h}4,l} \text{ for all } l \leq L \text{ (Appendix A.2)}\\
&\quad\approx_0^V C_{\mathsf{h}}[C[Q]]
\end{aligned}
$$

In the second step, in case the adversary makes an output on $c_{\mathsf{h}3,l}$ before outputting on $c_{\mathsf{h}1}$ (and inputting on $c_{\mathsf{h}2}$), this output blocks immediately in the process after elimination of communications on $c'_{\mathsf{h}3,l}, c'_{\mathsf{h}4,l}$ (because Q_{h} is not available yet); it succeeds in the process before elimination of communications on $c'_{\mathsf{h}3,l}, c'_{\mathsf{h}4,l}$, but the subsequent communication on $c'_{\mathsf{h}3,l}$ blocks (because Q'_{h} is not available yet). These two situations are indistinguishable. □

A.8 Proofs for Section 5.3

We write $\mathsf{AddIdx}(\widetilde{a}, Q)$ for the process obtained by adding indices $\widetilde{a}$ to all names of variables defined in Q and all names of events and tables in Q, and adding indices $\widetilde{a}$ at the beginning of all sequences of indices of channels in Q. This is just equivalent to renaming all variable, channel, event, and table names to distinct names for each value of $\widetilde{a}$. Similarly, we write $\mathsf{AddIdx}(\widetilde{a}, \mathit{corr})$ for the correspondence obtained by adding indices $\widetilde{a}$ to event names in corr.

Proof of Lemma 5. In this proof, we order the sequences $\widetilde{a}$ of the same length lexicographically and use $\prod$ for the indexed parallel composition. Let $Q_{\widetilde{a}} = \mathsf{AddIdx}(\widetilde{a}, Q)$ and $Q'_{\widetilde{a}} = \mathsf{AddIdx}(\widetilde{a}, Q')$. Let $V_{\widetilde{a}}$ be obtained by adding indices $\widetilde{a}$ to all variable names in V, and $V' = \bigcup_{\widetilde{a} \leq \widetilde{n}} V_{\widetilde{a}}$. We have $C'_{\mathsf{h}}[Q_{\widetilde{a}}] \approx_p^{V_{\widetilde{a}}} C'_{\mathsf{h}}[Q'_{\widetilde{a}}]$. By Lemma 4 applied with context $C_{\widetilde{a}} = \big(\prod_{\widetilde{a}' < \widetilde{a}} Q_{\widetilde{a}'}\big) \mid [\,] \mid \big(\prod_{\widetilde{a}' > \widetilde{a}, \widetilde{a}' \leq \widetilde{n}} Q'_{\widetilde{a}'}\big)$, there exists an evaluation context $C'_{\widetilde{a}}$ such that $C_{\mathsf{h}}[C_{\widetilde{a}}[Q_{\widetilde{a}}]] \approx_0^{V'} C'_{\widetilde{a}}[C'_{\mathsf{h}}[Q_{\widetilde{a}}]]$ and $C_{\mathsf{h}}[C_{\widetilde{a}}[Q'_{\widetilde{a}}]] \approx_0^{V'} C'_{\widetilde{a}}[C'_{\mathsf{h}}[Q'_{\widetilde{a}}]]$. Moreover, by adding context $C'_{\widetilde{a}}$, $C'_{\widetilde{a}}[C'_{\mathsf{h}}[Q_{\widetilde{a}}]] \approx_{p_{\widetilde{a}}}^{V'} C'_{\widetilde{a}}[C'_{\mathsf{h}}[Q'_{\widetilde{a}}]]$ where $p_{\widetilde{a}}(C_2, t_D) = p(C_2[C'_{\widetilde{a}}], t_D)$, so by transitivity, $C_{\mathsf{h}}[C_{\widetilde{a}}[Q_{\widetilde{a}}]] \approx_{p_{\widetilde{a}}}^{V'} C_{\mathsf{h}}[C_{\widetilde{a}}[Q'_{\widetilde{a}}]]$. This result allows us to replace one process $Q_{\widetilde{a}}$ with $Q'_{\widetilde{a}}$. We use a hybrid argument to replace all processes $Q_{\widetilde{a}}$ with $Q'_{\widetilde{a}}$ for $\widetilde{a} \leq \widetilde{n}$: by transitivity again,

$$C_{\mathsf{h}}\big[\prod_{\widetilde{a} \leq \widetilde{n}} Q_{\widetilde{a}}\big] \approx_{p''}^{V'} C_{\mathsf{h}}\big[\prod_{\widetilde{a} \leq \widetilde{n}} Q'_{\widetilde{a}}\big]$$

where

$$p''(C_2, t_D) = \sum_{\widetilde{a} \leq \widetilde{n}} p_{\widetilde{a}}(C_2, t_D) = \sum_{\widetilde{a} \leq \widetilde{n}} p(C_2[C'_{\widetilde{a}}], t_D)\,.$$

Let C be an acceptable evaluation context for $C_{\mathsf{h}}[Q_!]$ and $C_{\mathsf{h}}[Q'_!]$ with public variables V. We let C_1 be obtained from C by renaming the variables $y[\widetilde{a}, \widetilde{b}]$ into $y_{\widetilde{a}}[\widetilde{b}]$ for $y \in V$. Let $C_2 = C_1[\textbf{newChannel } c; (!^{\widetilde{i} \leq \widetilde{n}} c'[\widetilde{i}](x : T_{\mathsf{sid}}); \textbf{find } \widetilde{u} = \widetilde{i}' \leq \widetilde{n} \textbf{ suchthat defined}(x[\widetilde{i}'], x'[\widetilde{i}']) \wedge x = x[\widetilde{i}'] \textbf{ then yield else let } x' = \mathsf{cst} \textbf{ in } \overline{c[\widetilde{i}]}\langle\rangle \mid [\,])]$. We define f by $f(C) = C_2$.

We establish a correspondence between the traces of $C[C_{\mathsf{h}}[Q_!]]$ and the traces of $C_2[C_{\mathsf{h}}[\prod_{\widetilde{a}\leq\widetilde{n}} Q_{\widetilde{a}}]]$: we eliminate communications on c and map variables $y[\widetilde{a},\widetilde{b}]$ to $y_{\widetilde{a}}[\widetilde{b}]$ for $y \in \mathsf{var}(Q)$ and table entries $Tbl(a',\widetilde{b})$ to $Tbl_{\widetilde{a}}(\widetilde{b})$ for tables Tbl of Q where $\widetilde{a}$ is the index such that $x[\widetilde{a}] = a'$ and $x'[\widetilde{a}]$ is defined. (The structure of the **find** in C_2 guarantees that there exists exactly one such $\widetilde{a}$: $x'[\widetilde{a}]$ is defined when the process is executed with $x[\widetilde{a}] = a'$ for the first time. The processes Q and Q' do not contain events, so events are not affected by the correspondence. The indices of channels are the same in both processes.) Therefore, $\Pr[C[C_{\mathsf{h}}[Q_!]] : D] = \Pr[C_2[C_{\mathsf{h}}[\prod_{\widetilde{a}\leq\widetilde{n}} Q_{\widetilde{a}}]] : D]$, so $C_{\mathsf{h}}[Q_!] \overset{\sim}{\to}{}^{V,V'}_{f,0} C_{\mathsf{h}}[\prod_{\widetilde{a}\leq\widetilde{n}} Q_{\widetilde{a}}]$. Similarly, $C_{\mathsf{h}}[Q'_!] \overset{\sim}{\to}{}^{V,V'}_{f,0} C_{\mathsf{h}}[\prod_{\widetilde{a}\leq\widetilde{n}} Q'_{\widetilde{a}}]$. From $C_{\mathsf{h}}[\prod_{\widetilde{a}\leq\widetilde{n}} Q_{\widetilde{a}}] \approx^{V'}_{p''} C_{\mathsf{h}}[\prod_{\widetilde{a}\leq\widetilde{n}} Q'_{\widetilde{a}}]$ and these two properties, we conclude by Lemma 2 that

$$C_{\mathsf{h}}[Q_!] \approx^{V}_{p'''} C_{\mathsf{h}}[Q'_!]$$

where $p'''(C,t_D) = p''(C_2,t_D) = \sum_{\widetilde{a}\leq\widetilde{n}} p(C_2[C'_{\widetilde{a}}],t_D)$. Considering that the runtime of C_2 is about the same as the runtime of C, the context $C_2[C'_{\widetilde{a}}]$ runs in time at most $t_C + (\prod \widetilde{n} - 1) \times \max(t_Q, t_{Q'})$, calls the l-th hash oracle at most $n'_{\mathsf{h},l} = n_{\mathsf{h},l} + (\prod \widetilde{n} - 1) \times \max(n_{\mathsf{h},l,Q}, n_{\mathsf{h},l,Q'})$ times, and its other parameters are the same as those of C. Therefore, $p'''(C,t_D) \leq \prod \widetilde{n} \times p(C',t_D) = p'(C,t_D)$. □

Proof of Lemma 6, Property 1. Let Q° and $Q^\circ_!$ be obtained from Q and $Q_!$ respectively by removing all events. By Lemma 7, we have $C'_{\mathsf{h}}[Q^\circ] \mid R^0_x \approx^V_{p_1} C'_{\mathsf{h}}[Q^\circ] \mid R^1_x$ where $p_1(C,t_D) = p(C + t_D)$.

Let $Q_{\widetilde{a}} = \mathsf{AddIdx}(\widetilde{a}, Q^\circ)$, $R^0_{x,\widetilde{a}} = \mathsf{AddIdx}(\widetilde{a}, R^0_x)$, and $R^1_{x,\widetilde{a}} = \mathsf{AddIdx}(\widetilde{a}, R^1_x)$. Let $V_{\widetilde{a}}$ be obtained by adding indices $\widetilde{a}$ to all variable names in V, and $V' = \bigcup_{\widetilde{a}\leq\widetilde{n}} V_{\widetilde{a}}$. We have

$$C'_{\mathsf{h}}[Q_{\widetilde{a}}] \mid R^0_{x,\widetilde{a}} \approx^{V_{\widetilde{a}}}_{p_1} C'_{\mathsf{h}}[Q_{\widetilde{a}}] \mid R^1_{x,\widetilde{a}} \tag{8}$$

By Lemma 4 applied with the context $C_{\widetilde{a}} = [\,] \mid \left(\prod_{\widetilde{a}'\leq\widetilde{n},\widetilde{a}'\neq\widetilde{a}} Q_{\widetilde{a}'}\right)$, there exists an evaluation context $C'_{\widetilde{a}}$ such that $C_{\mathsf{h}}[C_{\widetilde{a}}[Q_{\widetilde{a}}]] \approx^{V'}_0 C'_{\widetilde{a}}[C'_{\mathsf{h}}[Q_{\widetilde{a}}]]$, so by adding context $[\,] \mid R^0_{x,\widetilde{a}}$,

$$\begin{aligned} C_{\mathsf{h}}[C_{\widetilde{a}}[Q_{\widetilde{a}}]] \mid R^0_{x,\widetilde{a}} &\approx^{V'}_0 C'_{\widetilde{a}}[C'_{\mathsf{h}}[Q_{\widetilde{a}}]] \mid R^0_{x,\widetilde{a}} \\ &\approx^{V'}_0 C'_{\widetilde{a}}[C'_{\mathsf{h}}[Q_{\widetilde{a}}] \mid R^0_{x,\widetilde{a}}] \end{aligned}$$

since the channels restricted by $C'_{\widetilde{a}}$ do not occur in $R^0_{x,\widetilde{a}}$, and similarly

$$C_{\mathsf{h}}[C_{\widetilde{a}}[Q_{\widetilde{a}}]] \mid R^1_{x,\widetilde{a}} \approx^{V'}_0 C'_{\widetilde{a}}[C'_{\mathsf{h}}[Q_{\widetilde{a}}] \mid R^1_{x,\widetilde{a}}]$$

Moreover, by adding context $C'_{\widetilde{a}}$ to (8),

$$C'_{\widetilde{a}}[C'_{\mathsf{h}}[Q_{\widetilde{a}}] \mid R^0_{x,\widetilde{a}}] \approx^{V'}_{p_{\widetilde{a}}} C'_{\widetilde{a}}[C'_{\mathsf{h}}[Q_{\widetilde{a}}] \mid R^1_{x,\widetilde{a}}]$$

where $p_{\widetilde{a}}(C_2,t_D) = p_1(C_2[C'_{\widetilde{a}}],t_D)$, so by transitivity,

$$C_{\mathsf{h}}[C_{\widetilde{a}}[Q_{\widetilde{a}}]] \mid R^0_{x,\widetilde{a}} \approx^{V'}_{p_{\widetilde{a}}} C_{\mathsf{h}}[C_{\widetilde{a}}[Q_{\widetilde{a}}]] \mid R^1_{x,\widetilde{a}}$$

Let $C''_{\widetilde{a}} = \left(\prod_{\widetilde{a}'<\widetilde{a}} R^0_{x,\widetilde{a}'}\right) \mid [\,] \mid \left(\prod_{\widetilde{a}'>\widetilde{a},\widetilde{a}'\leq\widetilde{n}} R^1_{x,\widetilde{a}'}\right)$. By adding context $C''_{\widetilde{a}}$, we obtain

$$C_{\mathsf{h}}\left[\prod_{\widetilde{a}\leq\widetilde{n}} Q_{\widetilde{a}}\right] \mid C''_{\widetilde{a}}[R^0_{x,\widetilde{a}}] \approx^{V'}_{p'_a} C_{\mathsf{h}}\left[\prod_{\widetilde{a}\leq\widetilde{n}} Q_{\widetilde{a}}\right] \mid C''_{\widetilde{a}}[R^1_{x,\widetilde{a}}]$$

where $p'_{\widetilde{a}}(C,t_D) = p_{\widetilde{a}}(C[C''_{\widetilde{a}}],t_D) = p_1(C[C''_{\widetilde{a}}[C'_{\widetilde{a}}]],t_D) = p(C[C''_{\widetilde{a}}[C'_{\widetilde{a}}]] + t_D)$. By transitivity,

$$C_{\mathsf{h}}\left[\prod_{\widetilde{a}\leq\widetilde{n}} Q_{\widetilde{a}}\right] \mid \left(\prod_{\widetilde{a}\leq\widetilde{n}} R^0_{x,\widetilde{a}}\right) \approx^{V'}_{p''} C_{\mathsf{h}}\left[\prod_{\widetilde{a}\leq\widetilde{n}} Q_{\widetilde{a}}\right] \mid \left(\prod_{\widetilde{a}\leq\widetilde{n}} R^1_{x,\widetilde{a}}\right)$$

where $p''(C, t_D) = \sum_{\widetilde{a} \leq \widetilde{n}} p'_{\widetilde{a}}(C, t_D)$.

Given a process Q and a replication index i that does not occur in Q, we write $\mathsf{AddIdx}_1(i \leq n, Q)$ for the process obtained by adding index i at the beginning of each sequence of indices of channels in inputs and outputs, at the beginning of each event, at the beginning of the indices of each variable defined in Q (implicit when current replication indices are omitted), and at the beginning of each insertion in a table, and adding the test $= i$ at the beginning of each **get** in a table. We define $\mathsf{AddRepl}(i \leq n, Q) = !^{i \leq n}\mathsf{AddIdx}_1(i \leq n, Q)$ and $\mathsf{AddRepl}(\widetilde{i} \leq \widetilde{n}, Q) = \mathsf{AddRepl}(i_1 \leq n_1, \dots \mathsf{AddRepl}(i_m \leq n_m, Q))$ when $\widetilde{i} = i_1, \dots, i_m$ and $\widetilde{n} = n_1, \dots, n_m$.

Using the same function f and trace correspondence as in Lemma 5, we show that

$$C_{\mathsf{h}}[Q^\circ_!] \mid \mathsf{AddRepl}(\widetilde{i} \leq \widetilde{n}, R^0_x) \xrightarrow{V,V'}_{f,0} C_{\mathsf{h}}\big[\prod_{\widetilde{a} \leq \widetilde{n}} Q_{\widetilde{a}}\big] \mid \big(\prod_{\widetilde{a} \leq \widetilde{n}} R^0_{x,\widetilde{a}}\big)$$
$$C_{\mathsf{h}}[Q^\circ_!] \mid \mathsf{AddRepl}(\widetilde{i} \leq \widetilde{n}, R^1_x) \xrightarrow{V,V'}_{f,0} C_{\mathsf{h}}\big[\prod_{\widetilde{a} \leq \widetilde{n}} Q_{\widetilde{a}}\big] \mid \big(\prod_{\widetilde{a} \leq \widetilde{n}} R^1_{x,\widetilde{a}}\big)$$

so by Lemma 2, we have

$$C_{\mathsf{h}}[Q^\circ_!] \mid \mathsf{AddRepl}(\widetilde{i} \leq \widetilde{n}, R^0_x) \approx^V_{p'_1} C_{\mathsf{h}}[Q^\circ_!] \mid \mathsf{AddRepl}(\widetilde{i} \leq \widetilde{n}, R^1_x) \tag{9}$$

where $p'_1(C, t_D) = p''(f(C), t_D)$.

Suppose that the variable x has type T and is defined under replications $!^{\widetilde{i'} \leq \widetilde{n'}}$ in Q. Let c'_s be a fresh channel. Let

$$\begin{aligned}C_2 = {} & \mathbf{newChannel}\ c_s;\\ & (!^{i'_s \leq n'_s} c'_s[i'_s](\widetilde{u} \leq \widetilde{n}, \widetilde{u}' \leq \widetilde{n}'); \overline{c_s[\widetilde{u}, i'_s]}\langle \widetilde{u}' \rangle; c_s[\widetilde{u}, i'_s](x' : T); \overline{c'_s[i'_s]}\langle x' \rangle \mid [\,])\end{aligned}$$

Let

$$\begin{aligned}
{R'_x}^0 &= !^{i'_s \leq n'_s} c'_s[i'_s](\widetilde{u} \leq \widetilde{n}, \widetilde{u}' \leq \widetilde{n}'); \mathbf{if\ defined}(x[\widetilde{u}, \widetilde{u}']) \mathbf{\ then\ } \overline{c'_s[i'_s]}\langle x[\widetilde{u}, \widetilde{u}'] \rangle\\
{R'_x}^1 &= !^{i'_s \leq n'_s} c'_s[i'_s](\widetilde{u} \leq \widetilde{n}, \widetilde{u}' \leq \widetilde{n}'); \mathbf{if\ defined}(x[\widetilde{u}, \widetilde{u}']) \mathbf{\ then}\\
&\quad \mathbf{find}\ u'_{s1} = i'_{s1} \leq n'_s \mathbf{\ suchthat\ defined}(y[i'_{s1}], \widetilde{u}[i'_{s1}], \widetilde{u}'[i'_{s1}]) \wedge \widetilde{u}[i'_{s1}] = \widetilde{u} \wedge \widetilde{u}'[i'_{s1}] = \widetilde{u}'\\
&\quad \mathbf{then}\ \overline{c'_s[i'_s]}\langle y[u'_{s1}] \rangle\\
&\quad \mathbf{else\ new}\ y : T; \overline{c'_s[i'_s]}\langle y \rangle
\end{aligned}$$

be processes R^0_x and R^1_x associated to $C_{\mathsf{h}}[Q^\circ_!]$, using channel c'_s instead of c_s. We have

$$C_2[C_{\mathsf{h}}[Q^\circ_!] \mid \mathsf{AddRepl}(\widetilde{i} \leq \widetilde{n}, R^0_x)] \approx^V_0 C_{\mathsf{h}}[Q^\circ_!] \mid {R'_x}^0$$
$$C_2[C_{\mathsf{h}}[Q^\circ_!] \mid \mathsf{AddRepl}(\widetilde{i} \leq \widetilde{n}, R^1_x)] \approx^V_0 C_{\mathsf{h}}[Q^\circ_!] \mid {R'_x}^1$$

by eliminating communications on c_s (Appendix A.2). Moreover, by adding context C_2 to (9), we obtain

$$C_2[C_{\mathsf{h}}[Q^\circ_!] \mid \mathsf{AddRepl}(\widetilde{i} \leq \widetilde{n}, R^0_x)] \approx^V_{p'_1} C_2[C_{\mathsf{h}}[Q^\circ_!] \mid \mathsf{AddRepl}(\widetilde{i} \leq \widetilde{n}, R^1_x)]$$

(We ignore the very small runtime of C_2.) So by transitivity of $\approx$, we have

$$C_{\mathsf{h}}[Q^\circ_!] \mid {R'_x}^0 \approx^V_{p'_1} C_{\mathsf{h}}[Q^\circ_!] \mid {R'_x}^1$$

By Lemma 7, we conclude that $C_{\mathsf{h}}[Q_!]$ preserves the secrecy of x with public variables V up to probability p', with $p'(C) = p'_1(C, t_D) = p''(f(C), t_D) = \sum_{\widetilde{a} \leq \widetilde{n}} p'_{\widetilde{a}}(f(C), t_D) =$

$\sum_{\widetilde{a}\leq\widetilde{n}} p(f(C)[C'''_{\widetilde{a}}[C'_{\widetilde{a}}]] + t_D)$. Since D is true when an event is executed, its runtime t_D can be neglected. Moreover, the context $f(C)[C'''_{\widetilde{a}}[C'_{\widetilde{a}}]]$ runs in time at most $t_C + (\prod \widetilde{n} - 1)t_Q$, calls the l-th hash oracle at most $n'_{\mathsf{h},l} = n_{\mathsf{h},l} + (\prod \widetilde{n} - 1)n_{\mathsf{h},l,Q}$ times where C calls the l-th hash oracle at most $n_{\mathsf{h},l}$ times, and its other parameters are the same as those of C. Therefore, the context $f(C)[C'''_{\widetilde{a}}[C'_{\widetilde{a}}]]$ has the same parameters as the context C' in the statement of the lemma, so $p'(C) = \prod \widetilde{n} \times p(C')$. □

Proof of Lemma 6, Property 2. Let $Q_{\widetilde{a}} = \mathsf{AddIdx}(\widetilde{a}, Q)$. Let $V_{\widetilde{a}}$ be obtained by adding indices $\widetilde{a}$ to all variable names in V, and $V' = \bigcup_{\widetilde{a}\leq\widetilde{n}} V_{\widetilde{a}}$. Let $corr_{\widetilde{a}} = \mathsf{AddIdx}(\widetilde{a}, corr)$. Let $corr' = \mathsf{AddSid}(T_{\mathsf{sid}}, corr)$. Let C be an evaluation context acceptable for $C_{\mathsf{h}}[Q_!]$ with public variables V that does not contain events used by $corr'$. Let C_1 be obtained from C by renaming the variables $y[\widetilde{a}, \widetilde{b}]$ to variables $y_{\widetilde{a}}[\widetilde{b}]$ for $y \in V$ and $\widetilde{a} \leq \widetilde{n}$. Let

$$\begin{aligned}C_2 = C_1[&\mathbf{newChannel}\ c; (!^{\widetilde{i}\leq\widetilde{n}} c'[\widetilde{i}](x : T_{\mathsf{sid}});\\ &\mathbf{find}\ \widetilde{u} = \widetilde{i}' \leq \widetilde{n}\ \mathbf{suchthat}\ \mathbf{defined}(x[\widetilde{i}'], x'[\widetilde{i}']) \wedge x = x[\widetilde{i}']\\ &\mathbf{then\ yield}\\ &\mathbf{else\ let}\ x' = \mathsf{cst}\ \mathbf{in}\ \overline{c[\widetilde{i}]}\langle\rangle \mid [\,])]\end{aligned}$$

Let $C_{\widetilde{a}} = \left(\prod_{\widetilde{a}'<\widetilde{n}, \widetilde{a}'\neq\widetilde{a}} Q_{\widetilde{a}'}\right) \mid [\,]$. We have

$$\begin{aligned}\mathsf{Adv}^{corr'}_{C_{\mathsf{h}}[Q_!]}(C) &= \Pr[C[C_{\mathsf{h}}[Q_!]] : \neg corr']\\ &= \Pr\left[C_2\left[C_{\mathsf{h}}\left[\prod\nolimits_{\widetilde{a}\leq\widetilde{n}} Q_{\widetilde{a}}\right]\right] : \neg\forall \widetilde{a} \leq \widetilde{n}, corr_{\widetilde{a}}\right]\end{aligned}$$

This equality of probabilities is shown by establishing a correspondence between the traces of $C[C_{\mathsf{h}}[Q_!]]$ and the traces of $C_2[C_{\mathsf{h}}[\prod_{\widetilde{a}\leq\widetilde{n}} Q_{\widetilde{a}}]]$: we eliminate communications on c and map variables $y[\widetilde{a}, \widetilde{b}]$ to $y_{\widetilde{a}}[\widetilde{b}]$ for $y \in \mathsf{var}(Q)$, table entries $Tbl(a', \widetilde{b})$ to $Tbl_{\widetilde{a}}(\widetilde{b})$ for tables Tbl of Q, and events $e(a', \widetilde{b})$ to events $e_{\widetilde{a}}(\widetilde{b})$ for events e that occur in $corr'$ (all other events can be ignored), where $\widetilde{a}$ is the index such that $x[\widetilde{a}] = a'$ and $x'[\widetilde{a}]$ is defined. Therefore, we have

$$\begin{aligned}\mathsf{Adv}^{corr'}_{C_{\mathsf{h}}[Q_!]}(C) &\leq \sum_{\widetilde{a}\leq\widetilde{n}} \Pr\left[C_2\left[C_{\mathsf{h}}\left[\prod\nolimits_{\widetilde{a}\leq\widetilde{n}} Q_{\widetilde{a}}\right]\right] : \neg corr_{\widetilde{a}}\right]\\ &\leq \sum_{\widetilde{a}\leq\widetilde{n}} \Pr[C_2[C_{\mathsf{h}}[C_{\widetilde{a}}[Q_{\widetilde{a}}]]] : \neg corr_{\widetilde{a}}]\end{aligned}$$

By Lemma 4 applied with context $C_{\widetilde{a}}$, there exists an evaluation context $C'_{\widetilde{a}}$ such that $C_{\mathsf{h}}[C_{\widetilde{a}}[Q_{\widetilde{a}}]] \approx_0^{V'} C'_{\widetilde{a}}[C'_{\mathsf{h}}[Q_{\widetilde{a}}]]$, so

$$\begin{aligned}\mathsf{Adv}^{corr'}_{C_{\mathsf{h}}[Q_!]}(C) &\leq \sum_{\widetilde{a}\leq\widetilde{n}} \Pr[C_2[C'_{\widetilde{a}}[C'_{\mathsf{h}}[Q_{\widetilde{a}}]]] : \neg corr_{\widetilde{a}}]\\ &\leq \sum_{\widetilde{a}\leq\widetilde{n}} p(C_2[C'_{\widetilde{a}}])\end{aligned}$$

since $C_2[C'_{\widetilde{a}}]$ does not contain events with indices $\widetilde{a}$, so does not contain events used by $corr_{\widetilde{a}}$. Moreover, the context $C_2[C'_{\widetilde{a}}]$ runs in time at most $t_C + (\prod \widetilde{n} - 1)t_Q$, calls the l-th hash oracle at most $n'_{\mathsf{h},l} = n_{\mathsf{h},l} + (\prod \widetilde{n} - 1)n_{\mathsf{h},l,Q}$ times, and its other parameters are the same as those of C. Therefore, the context $C_2[C'_{\widetilde{a}}]$ has the same parameters as the context C' in the statement of the lemma, so $\mathsf{Adv}^{corr'}_{C_{\mathsf{h}}[Q_!]}(C) \leq \prod \widetilde{n} \times p(C') = p'(C)$. Hence, $C_{\mathsf{h}}[Q_!]$ satisfies $corr' = \mathsf{AddSid}(T_{\mathsf{sid}}, corr)$ with public variables V up to probability p'. □

A.9 Proof for Section 5.4

Proof of Theorem 2. Let us first introduce general notations. Given a process Q that makes all its inputs and outputs on distinct channels with indices the current replication indices, let $\mathsf{ch}_{\mathsf{in}}(Q)$ (resp. $\mathsf{ch}_{\mathsf{out}}(Q)$) be the channels on which Q performs its inputs (resp. outputs), and $\mathsf{ch}(Q) = \mathsf{ch}_{\mathsf{in}}(Q) \cup \mathsf{ch}_{\mathsf{out}}(Q)$. For each channel c in $\mathsf{ch}(Q)$, let fr_c be a fresh channel and x_c and x'_c be distinct fresh variables. Let $\mathsf{ch}'(Q) = \{\mathsf{fr}_c \mid c \in \mathsf{ch}(Q)\}$. In Q, an input or output on channel c is under $!^{\widetilde{i}_c \leq \widetilde{n}_c}$, and has type T_c. We define two relay processes as follows:

$$\begin{aligned}
\mathsf{Relay}(Q, \widetilde{M}, \widetilde{M'}) = {} & \textstyle\prod_{c \in \mathsf{ch}_{\mathsf{in}}(Q)} !^{\widetilde{i}_c \leq \widetilde{n}_c} \mathsf{fr}_c[\widetilde{M}, \widetilde{i}_c](x_c : \mathsf{T}_c); \overline{c[\widetilde{M'}, \widetilde{i}_c]}\langle x_c \rangle \mid \\
& \textstyle\prod_{c \in \mathsf{ch}_{\mathsf{out}}(Q)} !^{\widetilde{i}_c \leq \widetilde{n}_c} c[\widetilde{M'}, \widetilde{i}_c](x_c : \mathsf{T}_c); \overline{\mathsf{fr}_c[\widetilde{M}, \widetilde{i}_c]}\langle x_c \rangle \\
\mathsf{Relay}'(Q, \widetilde{M}) = {} & \textstyle\prod_{c \in \mathsf{ch}_{\mathsf{in}}(Q)} !^{\widetilde{i}_c \leq \widetilde{n}_c} c[\widetilde{M}, \widetilde{i}_c](x'_c : \mathsf{T}_c); \overline{\mathsf{fr}_c[\widetilde{M}, \widetilde{i}_c]}\langle x'_c \rangle \mid \\
& \textstyle\prod_{c \in \mathsf{ch}_{\mathsf{out}}(Q)} !^{\widetilde{M} \leq \widetilde{n}'} !^{\widetilde{i}_c \leq \widetilde{n}_c} \mathsf{fr}_c[\widetilde{M}, \widetilde{i}_c](x'_c : \mathsf{T}_c); \overline{c[\widetilde{M}, \widetilde{i}_c]}\langle x'_c \rangle
\end{aligned}$$

The goal of these relay processes is to renumber the first channel indices in Q from $\widetilde{M'}$ to $\widetilde{M}$. However, to avoid confusions between channels before renumbering and channels after renumbering, we introduce fresh channels. So the relay process $\mathsf{Relay}(Q, \widetilde{M}, \widetilde{M'})$ performs the renumbering and forwards messages on channels fr_c for $c \in \mathsf{ch}_{\mathsf{in}}(Q)$ to c and forward the replies on channels $c \in \mathsf{ch}_{\mathsf{out}}(Q)$ back on fr_c. The relay process $\mathsf{Relay}'(Q, \widetilde{M})$ just performs the inverse renaming of channels: it forwards messages on channels $c \in \mathsf{ch}_{\mathsf{in}}(Q)$ to fr_c and forwards the replies on channels fr_c for $c \in \mathsf{ch}_{\mathsf{out}}(Q)$ back on c, so that after applying both relay processes, the messages are exchanged on channels in $\mathsf{ch}(Q)$ as in Q. The process $\mathsf{Relay}'(Q, \widetilde{M})$ does not renumber channel indices. We use the same notations for a context C instead of a process Q.

We can now start the proof itself. Let $\widetilde{u}$, $\widetilde{u}'$, and $\widetilde{u}''$ be fresh variables. In particular, they are not in V_1. Let

$$\begin{aligned}
C_5 = {} & \textbf{newChannel } \mathsf{ch}'(Q_{2B}); (!^{\widetilde{i}' \leq \widetilde{n}'} \mathsf{Relay}'(Q_{2B}, \widetilde{i}') \mid \\
& \quad \textbf{newChannel } c'_1, c_2, \mathsf{ch}(Q_{2B}); \\
& \quad (C[\textbf{event } e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \\
& \qquad\quad \textbf{find } \widetilde{u}'' = \widetilde{i}''' \leq \widetilde{n} \textbf{ suchthat defined}(\widetilde{msg}_A[\widetilde{i}'''], k'_A[\widetilde{i}''']) \wedge \\
& \qquad\qquad \mathsf{sid}(\widetilde{msg}_A) = \mathsf{sid}(\widetilde{msg}_A[\widetilde{i}''']) \textbf{ then yield else} \\
& \qquad\quad \textbf{let } k'_A = k_A \textbf{ in } \overline{c'_1[\widetilde{i}]}\langle \mathsf{sid}(\widetilde{msg}_A) \rangle; c_2[\widetilde{i}](); \overline{c'_A[\widetilde{i}]}\langle M_A \rangle; Q_{1A}, \\
& \qquad \textbf{event } e_B(\mathsf{sid}(\widetilde{msg}_B), k_B); \\
& \qquad\quad \textbf{find } \widetilde{u} = \widetilde{i}'' \leq \widetilde{n} \textbf{ suchthat defined}(\widetilde{msg}_A[\widetilde{i}''], k'_A[\widetilde{i}'']) \wedge \\
& \qquad\qquad \mathsf{sid}(\widetilde{msg}_A[\widetilde{i}'']) = \mathsf{sid}(\widetilde{msg}_B) \wedge \mathsf{fresh}(\widetilde{i}'', \widetilde{u}) \\
& \qquad\quad \textbf{then } \overline{c'_B[\widetilde{i}']}\langle M_B \rangle; (Q_{1B} \mid \mathsf{Relay}(Q_{2B}, \widetilde{i}', \widetilde{u}))] \\
& \quad \mid []))
\end{aligned}$$

where $\mathsf{fresh}(\widetilde{i}'', \widetilde{u}) = \textbf{find } \widetilde{u}' = \widetilde{i}''' \leq \widetilde{n}' \textbf{ suchthat defined}(\widetilde{u}[\widetilde{i}''']) \wedge \widetilde{u}[\widetilde{i}'''] = \widetilde{i}'' \textbf{ then } \mathsf{false} \textbf{ else } \mathsf{true}$. We have $\mathsf{fresh}(\widetilde{i}'', \widetilde{u})$ when $\widetilde{i}''$ was not used before, that is, it does not occur in the array $\widetilde{u}$.

Let

$$\begin{aligned}G_0 = C''_{\mathsf{h}}[C[&\mathbf{event}\ e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i});\\
&\quad\mathbf{find}\ \widetilde{u}'' = \widetilde{i}''' \leq \widetilde{n}\ \mathbf{suchthat}\ \mathbf{defined}(\widetilde{msg}_A[\widetilde{i}'''], k'_A[\widetilde{i}''']) \wedge\\
&\qquad\mathsf{sid}(\widetilde{msg}_A) = \mathsf{sid}(\widetilde{msg}_A[\widetilde{i}''']) \ \mathbf{then\ yield\ else}\\
&\quad\mathbf{let}\ k'_A = k_A\ \mathbf{in\ let}\ x = \mathsf{sid}(\widetilde{msg}_A)\ \mathbf{in}\\
&\quad\mathbf{new}\ k : T; \overline{c'_A[\widetilde{i}]}\langle M_A\rangle; (Q_{1A} \mid Q'_{2A}),\\
&\mathbf{event}\ e_B(\mathsf{sid}(\widetilde{msg}_B), k_B);\\
&\quad\mathbf{find}\ \widetilde{u} = \widetilde{i}'' \leq \widetilde{n}\ \mathbf{suchthat}\ \mathbf{defined}(\widetilde{msg}_A[\widetilde{i}''], k'_A[\widetilde{i}''], k[\widetilde{i}''], x[\widetilde{i}'']) \wedge\\
&\qquad\mathsf{sid}(\widetilde{msg}_A[\widetilde{i}'']) = \mathsf{sid}(\widetilde{msg}_B) \wedge \mathsf{fresh}(\widetilde{i}'', \widetilde{u})\ \mathbf{then}\\
&\quad\overline{c'_B[\widetilde{i}']}\langle M_B\rangle; (Q_{1B} \mid Q'_{2B}\{k[\widetilde{u}]/k, x[\widetilde{u}]/x\})]]\end{aligned}$$

and

$$\begin{aligned}G_1 = C''_{\mathsf{h}}[C[&\mathbf{event}\ e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i});\\
&\quad\mathbf{find}\ \widetilde{u}'' = \widetilde{i}''' \leq \widetilde{n}\ \mathbf{suchthat}\ \mathbf{defined}(\widetilde{msg}_A[\widetilde{i}'''], k[\widetilde{i}''']) \wedge\\
&\qquad\mathsf{sid}(\widetilde{msg}_A) = \mathsf{sid}(\widetilde{msg}_A[\widetilde{i}''']) \ \mathbf{then\ yield\ else}\\
&\quad\mathbf{new}\ k : T; \overline{c'_A[\widetilde{i}]}\langle M_A\rangle; (Q_{1A} \mid Q'_{2A}\{\mathsf{sid}(\widetilde{msg}_A)/x\}),\\
&\mathbf{event}\ e_B(\mathsf{sid}(\widetilde{msg}_B), k_B);\\
&\quad\mathbf{find}\ \widetilde{u} = \widetilde{i}'' \leq \widetilde{n}\ \mathbf{suchthat}\ \mathbf{defined}(\widetilde{msg}_A[\widetilde{i}''], k[\widetilde{i}'']) \wedge\\
&\qquad\mathsf{sid}(\widetilde{msg}_A[\widetilde{i}'']) = \mathsf{sid}(\widetilde{msg}_B) \wedge \mathsf{fresh}(\widetilde{i}'', \widetilde{u})\ \mathbf{then}\\
&\quad\overline{c'_B[\widetilde{i}']}\langle M_B\rangle; (Q_{1B} \mid Q'_{2B}\{k[\widetilde{u}]/k, \mathsf{sid}(\widetilde{msg}_B)/x\})]]\end{aligned}$$

The games G_0 and G_1 run the key exchange protocol followed by the protocol that uses the key, much like $S_{composed}$. However, in the participant A (after event e_A), they do not run the protocol that uses the key when the same session identifier has already been seen in a previous session (**find** $\widetilde{u}''$), and they generate a fresh key k instead of using the key provided by the key exchange protocol (**new** $k : T$). In the participant B (after event e_B), they get the key that has been generated in A with the same session identifier (**find** $\widetilde{u}$), and require that the key of a given session of A is reused at most once by B (condition $\mathsf{fresh}(\widetilde{i}'', \widetilde{u})$).

Let $Q'_2 = \mathsf{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c'_1, T_{\mathsf{sid}}, Q_2)$, so that $S_2 = C'_{\mathsf{h}}[Q'_2]$. We prove that $G_1 \xrightarrow{\sim}{}^{V_1, V_1 \cup \{\widetilde{u}\}}_{f',0} C''_{\mathsf{h}}[C_5[Q'_2]]$, using G_0 as intermediate game.

In the game $C''_{\mathsf{h}}[C_5[Q'_2]]$, the **find** $\widetilde{u}''$ in C_5 guarantees that the same value of $x = \mathsf{sid}(\widetilde{msg}_A)$ is never sent twice on channel c'_1, so the **find** introduced at the root of Q'_2 by $\mathsf{AddReplSid}$ never succeeds. Then we can remove this **find**, keeping only its **else** branch. We can also remove the assignment **let** $x' = \mathsf{cst}$ since x' is now unused. Let $Q''_{2B} = \mathsf{AddIdxSid}(\widetilde{i} \leq \widetilde{n}, x : T_{\mathsf{sid}}, Q_{2B})$ and $Q''_2 = !^{\widetilde{i} \leq \widetilde{n}} c'_1[\widetilde{i}](x : T_{\mathsf{sid}}); \mathbf{new}\ k : T; \overline{c_2[\widetilde{i}]}\langle\rangle; (Q'_{2A} \mid Q''_{2B})$. By the previous reasoning, we have $C''_{\mathsf{h}}[C_5[Q''_2]] \approx_0^{V_1 \cup \{\widetilde{u}\}} C''_{\mathsf{h}}[C_5[Q'_2]]$.

Second, we show $G_0 \xrightarrow{\sim}{}^{V_1, V_1 \cup \{\widetilde{u}\}}_{f',0} C''_{\mathsf{h}}[C_5[Q''_2]]$. To prove this property, let us consider an evaluation context C_6 acceptable for G_0 with public variables V_1. Let $f'(C_6) = C'_6$ be obtained from C_6 by replacing array accesses $y[\widetilde{M}, \widetilde{M}']$ with $y[\widetilde{u}[\widetilde{M}], \widetilde{M}']$, when $y \in V_1$ is defined in Q'_{2B}, $\widetilde{M}$ contains as many elements as $\widetilde{i}'$, and $\widetilde{M}'$ contains the other indices of y if any. The context C'_6 is an evaluation context acceptable for $C''_{\mathsf{h}}[C_5[Q''_2]]$ with public variables $V_1 \cup \{\widetilde{u}\}$. We establish a correspondence between the traces of $C_6[G_0]$ and those of $C'_6[C''_{\mathsf{h}}[C_5[Q''_2]]]$: we eliminate communications on the private channels c'_1, c_2, $\mathsf{ch}(Q_{2B})$, and $\mathsf{ch}'(Q_{2B})$ (Appendix A.2) and we renumber the variables of Q'_{2B}, replacing indices $\widetilde{a} \leq \widetilde{n}'$ in $C_6[G_0]$ with $\widetilde{u}[\widetilde{a}] \leq \widetilde{n}$ in $C'_6[C''_{\mathsf{h}}[C_5[Q''_2]]]$. (The condition $\mathsf{fresh}(\widetilde{i}'', \widetilde{u})$ guarantees that $\widetilde{u}$ never takes twice the same value, hence the function from $\widetilde{a}$ to $\widetilde{u}[\widetilde{a}]$ is injective. We exclude **defined** conditions in Q_2 to facilitate this renumbering.) In this correspondence between traces, an execution of $Q'_{2B} = \mathsf{AddIdxSid}(\widetilde{i}' \leq \widetilde{n}', x : T_{\mathsf{sid}}, Q_{2B})$ with replication indices $\widetilde{a}$ in $C_6[G_0]$ corresponds to an execution of $Q''_{2B} = \mathsf{AddIdxSid}(\widetilde{i} \leq \widetilde{n}, x : T_{\mathsf{sid}}, Q_{2B})$ with replication indices $\widetilde{u}[\widetilde{a}]$ in $C'_6[C''_{\mathsf{h}}[C_5[Q''_2]]]$. These two executions have the same values of k and x due to the substitution $k[\widetilde{u}]/k, x[\widetilde{u}]/x$ in G_0. They have the same value of other variables by the renumbering of variables, and produce the same events and table entries, since the events and table entries do not contain the replication indices $\widetilde{i}$ but the session identifier x. The channel indices are renumbered using the relay processes $\mathsf{Relay}(Q_{2B}, \widetilde{i}', \widetilde{u})$ and $\mathsf{Relay}'(Q_{2B}, \widetilde{i}')$. In $C''_{\mathsf{h}}[C_5[Q''_2]]$, in **find** $\widetilde{u}$, when $k'_A[\widetilde{i}'']$ is defined, the output on $c'_1[\widetilde{i}'']$ has been executed, so $x[\widetilde{i}'']$ and $k[\widetilde{i}'']$ are defined, the input on $c_2[\widetilde{i}'']$ has been executed, and the process Q''_{2B} is available with indices $\widetilde{u} = \widetilde{i}''$. Moreover, Q''_{2B} with these indices is not available earlier to the context, because the channels of Q''_{2B} are hidden by **newChannel** and are accessible only via the relay processes $\mathsf{Relay}(Q_{2B}, \widetilde{i}', \widetilde{u})$ and $\mathsf{Relay}'(Q_{2B}, \widetilde{i}')$. In G_0, the variables k'_A, x, and k are always defined at the same time. Hence, in both $C''_{\mathsf{h}}[C_5[Q''_2]]$ and G_0, when $k'_A[\widetilde{i}'']$ is defined, so are $k[\widetilde{i}'']$ and $x[\widetilde{i}'']$. We add them to the **defined** condition so that we can refer to $k[\widetilde{u}]$ and $x[\widetilde{u}]$ in $Q'_{2B}\{k[\widetilde{u}]/k, x[\widetilde{u}]/x\}$ in G_0.

Third, we show $G_1 \approx_0^{V_1} G_0$. Starting from G_0, we replace x with its value $\mathsf{sid}(\widetilde{msg}_A)$, and we note that $x[\widetilde{u}] = \mathsf{sid}(\widetilde{msg}_A[\widetilde{u}]) = \mathsf{sid}(\widetilde{msg}_B)$ by the condition of **find** $\widetilde{u}$, so we replace $x[\widetilde{u}]$ with $\mathsf{sid}(\widetilde{msg}_B)$. Since k'_A and x are defined at the same time as k, we replace k'_A and x with k in the **defined** conditions. Finally, we can remove the definitions of k'_A and x since they are now unused, and we obtain the game G_1.

By combining the previous three results, we have $G_1 \overset{\sim}{\rightarrow}{}_{f',0}^{V_1, V_1 \cup \{\widetilde{u}\}} C''_{\mathsf{h}}[C_5[Q'_2]]$. Let C_5° be obtained from C_5 by removing all events and G_1° be obtained from G_1 by removing all events of S_1. By Lemma 11, we have

$$G_1^\circ \overset{\sim}{\rightarrow}{}_{f',0}^{V_1, V_1 \cup \{\widetilde{u}\}} C''_{\mathsf{h}}[C_5^\circ[Q'_2]] \tag{10}$$

By Lemma 4, there exists an evaluation context C'_5 such that

$$C''_{\mathsf{h}}[C_5^\circ[Q'_2]] \approx_0^{V_1 \cup \{\widetilde{u}\}} C'_5[C'_{\mathsf{h}}[Q'_2]] = C'_5[S_2] \tag{11}$$

where the context C'_5 runs in time at most $t_{C_5^\circ} \leq t_1$, calls the l-th hash oracle in C'_{h} at most $n_{\mathsf{h},l,C_5^\circ} \leq n_{\mathsf{h},l,1}$ times, so $n'_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,1}$, and its other parameters are the same as those of C_5°.

The proof that $S^\circ_{composed} \overset{\sim}{\rightarrow}{}_{f,p_3}^{V_1, V_2} S_2$ then proceeds in two main steps:

1. First, we write the process G_1 above as an evaluation context around

$$S'_1 = C_{\mathsf{h}}[C[\textbf{event } e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \textbf{new } k'_A : T; \overline{c''_A[\widetilde{i}]}\langle(k'_A, M_A)\rangle; Q_{1A},$$
$$\textbf{event } e_B(\mathsf{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\widetilde{i}']}\langle M_B\rangle; Q_{1B}]]$$

   Let S''_1 be obtained by replacing **new** $k'_A : T$ with **let** $k'_A = k_A$ **in** in S'_1. Let G_2 be obtained by replacing **new** $k : T$ with **let** $k = k_A$ **in** in G_1. Let G_2°, $S_1'^\circ$, and $S_1''^\circ$ be obtained from G_2, S'_1, and S''_1 respectively by removing all events of S_1. Since S_1 preserves the secrecy of k'_A with public variables V ($k_A, k'_A \notin V$) up to probability p, by Lemma 9, $S_1'^\circ$ is indistinguishable from $S_1''^\circ$, so G_1° is indistinguishable from G_2°.

2. Second, we write G_2 as an evaluation context around S_1. Since S_1 satisfies the correspondences (2) and (3), so does G_2, so we obtain that, up to a small probability, the **find** $\widetilde{u}''$ in G_2 fails and the **find** $\widetilde{u}$ succeeds with $k[\widetilde{u}] = k_B$. From that, we show that G_2 is indistinguishable from $S_{composed}$, so by Lemma 11, G_2° is indistinguishable from $S^\circ_{composed}$.

We conclude by combining these results with (10) and (11). Let us detail these two steps.

Let $\widetilde{msg}'_A$ (resp. $\widetilde{msg}'_B$) be the sequence of variables corresponding to $\widetilde{msg}_A$ (resp. $\widetilde{msg}_B$), but using variables x_c of the relay process $\mathsf{Relay}(C, \emptyset, \emptyset)$ instead of variables of C. (We use $\emptyset$ for the empty sequence. In case a variable of $\widetilde{msg}_A$ is output by the output $\overline{c_A[\widetilde{i}]}\langle M_A\rangle$, M_A is equal to this variable, and $\widetilde{msg}'_A$ uses x_A instead of this variable. In the context C_1 below, x_A contains the value of M_A.) The goal of the relay process $\mathsf{Relay}(C, \emptyset, \emptyset)$ is to capture in variables $\widetilde{msg}'_A$, visible outside C, the content of $\widetilde{msg}_A$, which are variables internal to C but sent or received on public channels, and similarly for $\widetilde{msg}_B$. Since we use the relay process $\mathsf{Relay}(C, \emptyset, \emptyset)$, the external inputs and outputs are now done on the new channels fr_c for $c \in \mathsf{ch}(C)$, so we rename the channels $c \in \mathsf{ch}(C)$ to fr_c on the other side of the equivalence below.

Given a process Q that makes all its inputs and outputs on distinct channels with indices the current replication indices, let $\mathsf{ch}_{\mathsf{st}}(Q)$ be the channels on which Q performs its first inputs. For each channel c in $\mathsf{ch}_{\mathsf{st}}(Q)$, let fr_c be a fresh channel and x_c be a fresh variable. In Q, an input on channel c is under $!^{\widetilde{i}_c \leq \widetilde{n}_c}$, and has type T_c. We define a relay process as follows:

$$\mathsf{Relay}_{\mathsf{st}}(Q) = \prod_{c \in \mathsf{ch}_{\mathsf{st}}(Q)} !^{\widetilde{i}_c \leq \widetilde{n}_c} \mathsf{fr}_c[\widetilde{i}, \widetilde{i}_c](x_c : \mathsf{T}_c); \overline{c[\widetilde{i}, \widetilde{i}_c]}\langle x_c\rangle$$

The process $\mathsf{Relay}_{\mathsf{st}}(Q)$ relays messages received on fresh channels fr_c for $c \in \mathsf{ch}_{\mathsf{st}}(Q)$ to channel c. In the context C_1 below, we hide the channels in $\mathsf{ch}_{\mathsf{st}}(Q_{1A})$ and execute $\mathsf{Relay}_{\mathsf{st}}(Q_{1A})$ only when the **find** $\widetilde{u}''$ fails. As a result, the adversary against $C_1[S'_1]$ can access Q_{1A} (by sending messages on fr_{ch} for $c \in \mathsf{ch}_{\mathsf{st}}(Q_{1A})$) only when the **find** $\widetilde{u}''$ fails, similarly to what happens in G_1. We proceed in a similar way for Q_{1B}.

Suppose that M_A has type T_A and M_B has type T_B. Let c''_A be a fresh channel. Let k', x_A, x_B be fresh variables. Let

$$\begin{aligned}
C_1 = {}&\mathbf{newChannel}\ c''_A, c_B, \mathsf{ch}(C), \mathsf{ch}_{\mathsf{st}}(Q_{1A}), \mathsf{ch}_{\mathsf{st}}(Q_{1B});\\
&(\mathsf{Relay}(C, \emptyset, \emptyset) \mid\\
&!^{\widetilde{i} \leq \widetilde{n}} c''_A[\widetilde{i}]((k' : T, x_A : T_A));\\
&\quad \mathbf{find}\ \widetilde{u}'' = \widetilde{i}''' \leq \widetilde{n}\ \mathbf{suchthat}\ \mathbf{defined}(\widetilde{msg}'_A[\widetilde{i}'''], k[\widetilde{i}'''], \widetilde{msg}'_A[\widetilde{i}]) \wedge\\
&\qquad \mathsf{sid}(\widetilde{msg}'_A[\widetilde{i}]) = \mathsf{sid}(\widetilde{msg}'_A[\widetilde{i}''']) \ \mathbf{then\ yield\ else}\\
&\quad \mathbf{let}\ k = k'\ \mathbf{in}\ \overline{c'_A[\widetilde{i}]}\langle x_A\rangle; (\mathsf{Relay}_{\mathsf{st}}(Q_{1A}) \mid Q'_{2A}\{\mathsf{sid}(\widetilde{msg}'_A[\widetilde{i}])/x\}) \mid\\
&!^{\widetilde{i}' \leq \widetilde{n}'} c_B[\widetilde{i}'](x_B : T_B);\\
&\quad \mathbf{find}\ \widetilde{u} = \widetilde{i}'' \leq \widetilde{n}\ \mathbf{suchthat}\ \mathbf{defined}(\widetilde{msg}'_A[\widetilde{i}''], k[\widetilde{i}''], \widetilde{msg}'_B[\widetilde{i}']) \wedge\\
&\qquad \mathsf{sid}(\widetilde{msg}'_A[\widetilde{i}'']) = \mathsf{sid}(\widetilde{msg}'_B[\widetilde{i}']) \wedge \mathsf{fresh}(\widetilde{i}'', \widetilde{u})\ \mathbf{then}\\
&\quad \overline{c'_B[\widetilde{i}']}\langle x_B\rangle; (\mathsf{Relay}_{\mathsf{st}}(Q_{1B}) \mid Q'_{2B}\{k[\widetilde{u}]/k, \mathsf{sid}(\widetilde{msg}'_B[\widetilde{i}'])/x\}) \mid\\
&[\,])\\
Q'_1 = {}&C[\mathbf{event}\ e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \mathbf{new}\ k'_A : T; \overline{c''_A[\widetilde{i}]}\langle(k'_A, M_A)\rangle; Q_{1A},\\
&\quad \mathbf{event}\ e_B(\mathsf{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\widetilde{i}']}\langle M_B\rangle; Q_{1B}]
\end{aligned}$$

Let G_2 be obtained by replacing **new** $k : T$ with **let** $k = k_A$ **in** in G_1:

$$\begin{aligned}
G_2 = C''_{\mathsf{h}}[C[&\textbf{event } e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i});\\
&\quad\textbf{find } \widetilde{u}'' = \widetilde{i}''' \leq \widetilde{n} \textbf{ suchthat defined}(\widetilde{msg}_A[\widetilde{i}'''], k[\widetilde{i}''']) \wedge\\
&\quad\quad \mathsf{sid}(\widetilde{msg}_A) = \mathsf{sid}(\widetilde{msg}_A[\widetilde{i}''']) \textbf{ then yield else}\\
&\quad\textbf{let } k = k_A \textbf{ in } \overline{c'_A[\widetilde{i}]}\langle M_A\rangle; (Q_{1A} \mid Q'_{2A}\{\mathsf{sid}(\widetilde{msg}_A)/x\}),\\
&\textbf{event } e_B(\mathsf{sid}(\widetilde{msg}_B), k_B);\\
&\quad\textbf{find } \widetilde{u} = \widetilde{i}'' \leq \widetilde{n} \textbf{ suchthat defined}(\widetilde{msg}_A[\widetilde{i}''], k[\widetilde{i}'']) \wedge\\
&\quad\quad \mathsf{sid}(\widetilde{msg}_A[\widetilde{i}'']) = \mathsf{sid}(\widetilde{msg}_B) \wedge \mathsf{fresh}(\widetilde{i}'', \widetilde{u}) \textbf{ then}\\
&\quad\overline{c'_B[\widetilde{i}']}\langle M_B\rangle; (Q_{1B} \mid Q'_{2B}\{k[\widetilde{u}]/k, \mathsf{sid}(\widetilde{msg}_B)/x\})]]
\end{aligned}$$

Let Q''_1 be obtained by replacing **new** $k'_A : T$ with **let** $k'_A = k_A$ **in** in Q'_1. Let G°_2, ${Q'_1}^\circ$, and ${Q''_1}^\circ$ be obtained from G_2, Q'_1, and Q''_1 respectively by removing all events of S_1.

We have $G_1\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1A}) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1B})\} \approx_0^{V_1} C''_{\mathsf{h}}[C_1[Q'_1]]$, by eliminating communications on c''_A, c_B, $\mathsf{ch}(C)$, $\mathsf{ch}_{\mathsf{st}}(Q_{1A})$, and $\mathsf{ch}_{\mathsf{st}}(Q_{1B})$ (Appendix A.2), since the **find**s in C_1 have the same effect as the ones in G_1, because $\widetilde{msg}'_A$ and $\widetilde{msg}'_B$ contain the same value as $\widetilde{msg}_A$ and $\widetilde{msg}_B$ respectively, by construction. Since C_1 does not contain events of S_1, by Lemma 11,

$$G^\circ_1\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1A}) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1B})\} \approx_0^{V_1} C''_{\mathsf{h}}[C_1[{Q'_1}^\circ]] \tag{12}$$

Similarly, $G_2\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1A}) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1B})\} \approx_0^{V_1} C''_{\mathsf{h}}[C_1[Q''_1]]$ so by Lemma 11,

$$G^\circ_2\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1A}) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1B})\} \approx_0^{V_1} C''_{\mathsf{h}}[C_1[{Q''_1}^\circ]] \tag{13}$$

By Lemma 4, there exists an evaluation context C'_1 such that

$$C''_{\mathsf{h}}[C_1[{Q'_1}^\circ]] \approx_0^{V_1} C'_1[C_{\mathsf{h}}[{Q'_1}^\circ]] \tag{14}$$

$$C''_{\mathsf{h}}[C_1[{Q''_1}^\circ]] \approx_0^{V_1} C'_1[C_{\mathsf{h}}[{Q''_1}^\circ]] \tag{15}$$

where the context C'_1 runs in time at most $t_{C_1} \leq t_2$, calls the l-th hash oracle in C_{h} at most $n_{\mathsf{h},l,C_1} \leq n_{\mathsf{h},l,2}$ times, so $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$, and its other parameters are the same as those of C_1.

Since $S_1 = C_{\mathsf{h}}[Q_1]$ preserves the secrecy of k'_A with public variables V up to probability p, by Lemma 9, we have $C_{\mathsf{h}}[{Q'_1}^\circ] \approx_{p_0}^{V} C_{\mathsf{h}}[{Q''_1}^\circ]$ where $p_0(C_3, t_D) = p(C_3 + t_D)$. Therefore,

$$C'_1[C_{\mathsf{h}}[{Q'_1}^\circ]] \approx_{p_1}^{V_1} C'_1[C_{\mathsf{h}}[{Q''_1}^\circ]] \tag{16}$$

where $p_1(C_3, t_D) = p_0(C_3[C'_1], t_D) = p(C_3[C'_1] + t_D) = p(C'_3 + t_D)$ and the context C'_3 runs in time at most $t_{C_3} + t_2$, calls the l-th hash oracle at most $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$ times, and its other parameters are the same as those of C_3.

By combining (12), (14), (16), (15), and (13) by transitivity, we obtain

$$G^\circ_1\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1A}) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1B})\} \approx_{p_1}^{V_1} G^\circ_2\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1A}) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1B})\}$$

so by renaming channels

$$G^\circ_1 \approx_{p_1}^{V_1} G^\circ_2 \tag{17}$$

Introducing relay processes and renaming channels as above, we define

$$
\begin{aligned}
C_2 = {} & \mathbf{newChannel}\ c_A, c_B, \mathsf{ch}(C), \mathsf{ch}_{\mathsf{st}}(Q_{1A}), \mathsf{ch}_{\mathsf{st}}(Q_{1B});\\
& (\mathsf{Relay}(C, \emptyset, \emptyset) \mid\\
& !^{\widetilde{i} \leq \widetilde{n}} c_A[\widetilde{i}](x_A : T_A);\\
& \quad \mathbf{find}\ \widetilde{u}'' = \widetilde{i}''' \leq \widetilde{n}\ \mathbf{suchthat}\ \mathbf{defined}(\widetilde{msg}'_A[\widetilde{i}'''], k[\widetilde{i}'''], \widetilde{msg}'_A[\widetilde{i}]) \wedge\\
& \qquad \mathsf{sid}(\widetilde{msg}'_A[\widetilde{i}]) = \mathsf{sid}(\widetilde{msg}'_A[\widetilde{i}''']) \ \mathbf{then\ yield\ else}\\
& \quad \mathbf{if\ defined}(k'_A[\widetilde{i}])\ \mathbf{then\ let}\ k = k'_A[\widetilde{i}]\ \mathbf{in}\ \overline{c'_A[\widetilde{i}]}\langle x_A\rangle;\\
& \quad (\mathsf{Relay}_{\mathsf{st}}(Q_{1A}) \mid Q'_{2A}\{\mathsf{sid}(\widetilde{msg}'_A[\widetilde{i}])/x\}) \mid\\
& !^{\widetilde{i}' \leq \widetilde{n}'} c_B[\widetilde{i}'](x_B : T_B);\\
& \quad \mathbf{find}\ \widetilde{u} = \widetilde{i}'' \leq \widetilde{n}\ \mathbf{suchthat}\ \mathbf{defined}(\widetilde{msg}'_A[\widetilde{i}''], k[\widetilde{i}''], \widetilde{msg}'_B[\widetilde{i}']) \wedge\\
& \qquad \mathsf{sid}(\widetilde{msg}'_A[\widetilde{i}'']) = \mathsf{sid}(\widetilde{msg}'_B[\widetilde{i}']) \wedge \mathsf{fresh}(\widetilde{i}'', \widetilde{u})\ \mathbf{then}\\
& \quad \overline{c'_B[\widetilde{i}']}\langle x_B\rangle; (\mathsf{Relay}_{\mathsf{st}}(Q_{1B}) \mid Q'_{2B}\{k[\widetilde{u}]/k, \mathsf{sid}(\widetilde{msg}'_B[\widetilde{i}'])/x\}) \mid\\
& [\,])
\end{aligned}
$$

so that we have

$$G_2\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1A}) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1B})\} \approx_0^{V_1} C''_{\mathsf{h}}[C_2[Q_1]]$$

By Lemma 4, there exists an evaluation context C'_2 such that

$$C''_{\mathsf{h}}[C_2[Q_1]] \approx_0^{V_1} C'_2[C_{\mathsf{h}}[Q_1]] = C'_2[S_1]$$

where the context C'_2 runs in time at most $t_{C_2} \leq t_2$, calls the l-th hash oracle in C_{h} at most $n_{\mathsf{h},l,C_2} \leq n_{\mathsf{h},l,2}$ times, so $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$, and its other parameters are the same as those of C_2. Therefore,

$$G_2\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1A}) \cup \mathsf{ch}_{\mathsf{st}}(Q_{1B})\} \approx_0^{V_1} C'_2[S_1]$$

The process S_1 satisfies the correspondences (2) and (3) with public variables $V \cup \{k'_A\}$ up to probabilities p' and p'' respectively. Hence, by Lemma 1, the process $C'_2[S_1]$ satisfies these correspondences with public variables V_1 up to probabilities $p'_1(C_3, t_D) = p'(C_3[C'_2], t_D) = p'(C'_3, t_D)$ and $p''_1(C_3, t_D) = p''(C_3[C'_2], t_D) = p''(C'_3, t_D)$ respectively, where the context C'_3 runs in time at most $t_{C_3} + t_2$, calls the l-th hash oracle at most $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$ times, and its other parameters are the same as those of C_3, and so does G_2.

In G_2, for traces that satisfy these two correspondences, we have that event e_A is never executed twice with the same session identifier $\mathsf{sid}(\widetilde{msg}_A)$ because of the correspondence (3) so **find** $\widetilde{u}''$ always fails. It can then be removed, keeping only its **else** branch. Moreover, by (2), for each $\widetilde{i}'$ such that event $e_B(\mathsf{sid}(\widetilde{msg}_B[\widetilde{i}']), k_B[\widetilde{i}'])$ has been executed, there exists a distinct $\widetilde{i}$ such that $e_A(\mathsf{sid}(\widetilde{msg}_A[\widetilde{i}]), k_A[\widetilde{i}], \widetilde{i})$ has been executed, with $\mathsf{sid}(\widetilde{msg}_A[\widetilde{i}]) = \mathsf{sid}(\widetilde{msg}_B[\widetilde{i}'])$ and $k_A[\widetilde{i}] = k_B[\widetilde{i}']$. Therefore, the conditions of **find** $\widetilde{u}$ are satisfied with $\widetilde{i}'' = \widetilde{i}$, so **find** $\widetilde{u}$ succeeds. (Injectivity guarantees that we can always find a fresh $\widetilde{i}''$.) Furthermore, by (3), there is at most one $\widetilde{i}$ such that $e_A(\mathsf{sid}(\widetilde{msg}_A[\widetilde{i}]), k_A[\widetilde{i}], \widetilde{i})$ is executed with $\mathsf{sid}(\widetilde{msg}_A[\widetilde{i}]) = \mathsf{sid}(\widetilde{msg}_B[\widetilde{i}'])$, so we have $\widetilde{i}'' = \widetilde{i}$ and $k[\widetilde{u}] = k[\widetilde{i}''] = k[\widetilde{i}] = k_A[\widetilde{i}] = k_B[\widetilde{i}']$. Therefore, we can run $Q'_{2B}\{k_B[\widetilde{i}']/k, \mathsf{sid}(\widetilde{msg}_B)/x\}$

when **find** $\widetilde{u}$ succeeds. Since $\widetilde{u}$ is no longer used, we can remove **find** $\widetilde{u}$. Hence,

$$\begin{aligned}G_2 \approx^{V_1}_{p'_1+p''_1} C''_{\mathsf{h}}[C[&\textbf{event } e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \textbf{let } k = k_A \textbf{ in } \overline{c'_A[\widetilde{i}]}\langle M_A\rangle;\\ &(Q_{1A} \mid Q'_{2A}\{\mathsf{sid}(\widetilde{msg}_A)/x\}),\\ &\textbf{event } e_B(\mathsf{sid}(\widetilde{msg}_B), k_B); \overline{c'_B[\widetilde{i}']}\langle M_B\rangle; (Q_{1B} \mid Q'_{2B}\{k_B/k, \mathsf{sid}(\widetilde{msg}_B)/x\})]]\\ \approx^{V_1}_{p'_1+p''_1} S_{composed}&\end{aligned}$$

By Lemma 11, we have

$$G_2^\circ \approx^{V_1}_{p'_1+p''_1} S^\circ_{composed} \tag{18}$$

so by combining (18), (17), (10), and (11), we obtain

$$S^\circ_{composed} \approx^{V_1}_{p'_1+p''_1} G_2^\circ \approx^{V_1}_{p_1} G_1^\circ \mathrel{\overset{\sim}{\rightarrow}}^{V_1, V_1\cup\{\widetilde{u}\}}_{f',0} C''_{\mathsf{h}}[C_5^\circ[Q'_2]] \approx^{V_1\cup\{\widetilde{u}\}}_0 C'_5[S_2]$$

so $S^\circ_{composed} \mathrel{\overset{\sim}{\rightarrow}}^{V_1,V_1\cup\{\widetilde{u}\}}_{f',p_1+p'_1+p''_1} C'_5[S_2]$, so we obtain $S^\circ_{composed} \mathrel{\overset{\sim}{\rightarrow}}^{V_1,V_2}_{f,p_3} S_2$ by defining $f(C_3) = f'(C_3)[C'_5[\,]]$.

Furthermore, by Lemma 10, if $y \in V_2 \cap \mathsf{var}(Q_{2A})$, then f is secrecy-preserving for $y \mapsto (y, f_{\mathrm{sec}})$ where $f_{\mathrm{sec}}(C_3) = f'(C_3)[C'_5[\,]]$. Let us verify the assumptions of Lemma 10. First, we have $(\mathsf{var}(f'(C_3)) \cup \mathsf{var}(C'_5)) \cap \mathsf{var}(R_y) = \emptyset$. Indeed, $\mathsf{var}(C_3) \cap \mathsf{var}(R_y) = \emptyset$ because C_3 is an evaluation context acceptable for $S^\circ_{composed} \mid R_y$ with public variables $V_1 \setminus \{y\}$ and $(V_1 \setminus \{y\}) \cap \mathsf{var}(R_y) = \emptyset$ because the variables of R_y other than y are fresh. Moreover, $\mathsf{var}(C'_5) \cap \mathsf{var}(R_y) \subseteq \mathsf{var}(S_1) \cap \{y\} = \emptyset$ because $y \in \mathsf{var}(S_2)$ and S_1 and S_2 have no common variable, and the other variables are fresh: the variables of C'_5 are those of C_5° plus fresh variables by the construction done in Lemma 4, the variables of C_5° are those of S_1 plus the fresh variables $\widetilde{u}, \widetilde{u}'$, and the variables of R_y other than y are fresh. Second, let S_1° be obtained from S_1 by removing all events. We have $\mathsf{vardef}(f'(C_3)) \cap \mathsf{var}(C'_5) = \mathsf{vardef}(C_3) \cap \mathsf{var}(S_1^\circ) \subseteq \mathsf{vardef}(C_3) \cap \mathsf{var}(S^\circ_{composed} \mid R_y) = \emptyset$ because, as above, the variables of C'_5 are those of S_1° plus fresh variables. Third, $f'(C_3)$ and C'_5 do not use any common table because C_3 is an evaluation context acceptable for $S^\circ_{composed} \mid R_y$ with public variables $V_1 \setminus \{y\}$ so C_3 and $S^\circ_{composed} \mid R_y$ have no common table, the tables of $f'(C_3)$ are those of C_3, and the tables of C'_5 are those of C_5° by the construction done in Lemma 4, which are among those of $S^\circ_{composed}$.

This property does not apply to variables $y \in V_2 \cap \mathsf{var}(Q_{2B})$ because they are renumbered by f, so we perform a separate proof for such variables. Let $y \in V_2 \cap \mathsf{var}(Q_{2B})$. Suppose that S_2 preserves the secrecy of y with public variables $V_2 \setminus \{y\}$ up to probability p_2. By Lemma 1, $C'_5[S_2]$ preserves the secrecy of y with public variables $(V_2 \cup \mathsf{var}(C'_5)) \setminus \{y\}$ up to probability p'_2 such that $p'_2(C_5) = p_1(C_5[C'_5])$. Let R_y be the process used for testing secrecy of y in $S^\circ_{composed}$. Let R'_y be obtained by renumbering the variables R_y: $R'_y = f'(R_y)$, that is,

$$\begin{aligned}R'_y = c_{s0}();\, &\textbf{new } b : \mathit{bool}; \overline{c_{s0}}\langle\rangle;\\ &(!^{i_s \le n_s} c_s[i_s](\widetilde{u}_1 \le \widetilde{n}', \widetilde{u}_2 \le \widetilde{n}_2); \textbf{if defined}(y[\widetilde{u}[\widetilde{u}_1], \widetilde{u}_2]) \textbf{ then}\\ &\quad \textbf{if } b \textbf{ then } \overline{c_s[i_s]}\langle y[\widetilde{u}[\widetilde{u}_1], \widetilde{u}_2]\rangle \textbf{ else}\\ &\quad \textbf{find } u'_s = i'_s \le n_s \textbf{ suchthat defined}(y'[i'_s], \widetilde{u}_1[i'_s], \widetilde{u}_2[i'_s]) \wedge \widetilde{u}_1[i'_s] = \widetilde{u}_1 \wedge \widetilde{u}_2[i'_s] = \widetilde{u}_2\\ &\quad \textbf{then } \overline{c_s[i_s]}\langle y'[u'_s]\rangle \textbf{ else new } y' : T; \overline{c_s[i_s]}\langle y'\rangle\\ &\mid c'_s(b'); \textbf{if } b = b' \textbf{ then event\_abort } \mathsf{S} \textbf{ else event\_abort } \overline{\mathsf{S}})\end{aligned}$$

Let c''_s be a fresh channel, $\widetilde{u}'_1$, $\widetilde{u}'_2$, y'', $\widetilde{u}''_1$, $\widetilde{u}''_2$, y''' be fresh variables, and let us define relay processes:

$$\begin{aligned}\mathsf{Relay}_R = {} & !^{i_s \le n_s}\, c''_s[i_s](\widetilde{u}'_1 \le \widetilde{n}', \widetilde{u}'_2 \le \widetilde{n}_2); \textbf{if defined}(\widetilde{u}[\widetilde{u}'_1]) \textbf{ then}\\ & \overline{c_s[i_s]}\langle \widetilde{u}[\widetilde{u}'_1], \widetilde{u}'_2\rangle; c_s[i_s](y'' : T); \overline{c''_s[i_s]}\langle y''\rangle\\ \mathsf{Relay}'_R = {} & !^{i_s \le n_s}\, c_s[i_s](\widetilde{u}''_1 \le \widetilde{n}', \widetilde{u}''_2 \le \widetilde{n}_2); \overline{c''_s[i_s]}\langle \widetilde{u}''_1, \widetilde{u}''_2\rangle; c''_s[i_s](y''' : T); \overline{c_s[i_s]}\langle y'''\rangle\end{aligned}$$

The process Relay_R renumbers the queries for y like f' renumbers y. However, it uses channel c''_s instead of c_s, so we use process Relay'_R to revert to channel c_s. Let $C_R = \textbf{newChannel}\ c''_s; (\mathsf{Relay}'_R \mid \textbf{newChannel}\ c_s; (\mathsf{Relay}_R \mid [\,]))$. Let R''_y be the process used for testing secrecy of y in $C'_5[S_2]$. (The process R''_y differs from R_y because y has indices $\widetilde{i}' \le \widetilde{n}', \widetilde{i}_2 \le \widetilde{n}_2$ in $S^\circ_{composed}$, while it has indices $\widetilde{i} \le \widetilde{n}, \widetilde{i}_2 \le \widetilde{n}_2$ in $C'_5[S_2]$.) We have

$$C'_5[S_2] \mid R'_y \approx_0^{V_1} C_R[C'_5[S_2] \mid R''_y]$$

by eliminating communications on the private channels c_s and c''_s (Appendix A.2). The equality test $\widetilde{u}_1[i'_s] = \widetilde{u}_1$ performed by R'_y in the left-hand side becomes $\widetilde{u}[\widetilde{u}_1[i'_s]] = \widetilde{u}[\widetilde{u}_1]$ in the right-hand side because R''_y always receives $\widetilde{u}[\widetilde{u}_1]$ instead of $\widetilde{u}_1$. These two tests are equivalent, because $\widetilde{u}[\widetilde{i}']$ cannot have the same value for different values of $\widetilde{i}'$ due to the condition $\mathsf{fresh}(\widetilde{i}'', \widetilde{u})$.

Let C_0 be any evaluation context acceptable for $S^\circ_{composed} \mid R_y$ with public variables $V_1 \setminus \{y\}$. The context $f'(C_0)[C_R]$ is acceptable for $C'_5[S_2] \mid R''_y$ with public variables $(V_2 \cup \mathsf{var}(C'_5)) \setminus \{y\}$, by the definition of acceptable evaluation contexts [14, Definition 4]. Indeed,

$$\begin{aligned}&\mathsf{var}(f'(C_0)[C_R]) \cap \mathsf{var}(C'_5[S_2] \mid R''_y)\\ &\quad = \mathsf{var}(C_0) \cap (\mathsf{var}(C'_5) \cup \mathsf{var}(C'_{\mathsf{h}}) \cup \mathsf{var}(Q_2) \cup \{y\}) \qquad \text{because the other variables are fresh}\\ &\quad \subseteq \mathsf{var}(C_0) \cap (\mathsf{var}(C'_5) \cup ((\mathsf{var}(C'_{\mathsf{h}}) \cup \mathsf{var}(Q_2) \cup \{y\}) \cap V_1 \setminus \{y\})) && (19)\\ &\quad \subseteq \mathsf{var}(C_0) \cap (\mathsf{var}(C'_5) \cup (V_2 \setminus \{y\}))\\ &\quad \subseteq (\mathsf{var}(C'_5) \cup V_2) \setminus \{y\} && (20)\end{aligned}$$

The inclusion (19) holds because $\mathsf{var}(C_0) \cap (\mathsf{var}(C'_{\mathsf{h}}) \cup \mathsf{var}(Q_2) \cup \{y\}) \subseteq \mathsf{var}(C_0) \cap \mathsf{var}(S^\circ_{composed} \mid R_y) \subseteq V_1 \setminus \{y\}$ and (20) holds because $y \notin \mathsf{var}(C_0)$ by the inclusion $\mathsf{var}(C_0) \cap (\mathsf{var}(Q_2) \cup \{y\}) \subseteq V_1 \setminus \{y\}$ shown above. We also have $\mathsf{vardef}(f'(C_0)[C_R]) \cap ((V_2 \cup \mathsf{var}(C'_5)) \setminus \{y\}) \subseteq \mathsf{vardef}(C_0) \cap ((V_1 \cup \mathsf{var}(S^\circ_{composed} \mid R_y)) \setminus \{y\}) = \emptyset$ because the variables of $f'(C_0)[C_R]$ are those of C_0 plus fresh variables, $V_2 \subseteq V_1$, the variables of C'_5 are $\mathsf{var}(S^\circ_1) \subseteq \mathsf{var}(S^\circ_{composed})$ plus fresh variables, $\mathsf{vardef}(C_0) \cap (V_1 \setminus \{y\}) = \emptyset$, and $\mathsf{vardef}(C_0) \cap \mathsf{var}(S^\circ_{composed} \mid R_y) = \emptyset$. Moreover, $f'(C_0)[C_R]$ and $C'_5[S_2] \mid R''_y$ do not use any common table, because C_0 and $S^\circ_{composed} \mid R_y$ do not use any common table, the tables of $f'(C_0)[C_R]$ are those of C_0, and the tables of $C'_5[S_2] \mid R''_y$ are those of C°_5 and S_2, that is, of S_1 and S_2, so of $S^\circ_{composed}$. We have

$$\begin{aligned}&\Pr[C_0[S^\circ_{composed} \mid R_y] : \mathsf{S}] - \Pr[C_0[S^\circ_{composed} \mid R_y] : \overline{\mathsf{S}}]\\ &\quad \le |\Pr[C_0[S^\circ_{composed} \mid R_y] : \mathsf{S}] - \Pr[f'(C_0)[C'_5[S_2] \mid R'_y] : \mathsf{S}]| +\\ &\qquad (\Pr[f'(C_0)[C'_5[S_2] \mid R'_y] : \mathsf{S}] - \Pr[f'(C_0)[C'_5[S_2] \mid R'_y] : \overline{\mathsf{S}}]) +\\ &\qquad |\Pr[f'(C_0)[C'_5[S_2] \mid R'_y] : \overline{\mathsf{S}}] - \Pr[C_0[S^\circ_{composed} \mid R_y] : \overline{\mathsf{S}}]|\\ &\quad \le p_3(C_0[[\,] \mid R_y], t_{\mathsf{S}}) +\\ &\qquad (\Pr[f'(C_0)[C_R[C'_5[S_2] \mid R''_y]] : \mathsf{S}] - \Pr[f'(C_0)[C_R[C'_5[S_2] \mid R''_y]] : \overline{\mathsf{S}}]) +\\ &\qquad p_3(C_0[[\,] \mid R_y], t_{\overline{\mathsf{S}}})\end{aligned}$$

$$\begin{aligned}&\leq 2p_3(C_0[[\,] \mid R_y], t_{\mathsf{S}}) + p_2'(f'(C_0)[C_R])\\ &\leq 2p_3(C_0[[\,] \mid R_y], t_{\mathsf{S}}) + p_2(f_{\mathrm{sec}}(C_0))\end{aligned}$$

since $t_{\mathsf{S}} = t_{\overline{\mathsf{S}}}$ and $p_2'(f'(C_0)[C_R]) = p_2(f'(C)[C_R[C_5']]) = p_2(f_{\mathrm{sec}}(C_0))$ by defining $f_{\mathrm{sec}}(C_0) = f'(C_0)[C_R[C_5']]$. We conclude that $S_{composed}$ preserves the secrecy of y with public variables $V_1 \setminus \{y\}$ up to probability p'', where $p''(C_0) = 2p_3(C_0[[\,] \mid R_y], t_{\mathsf{S}}) + p_2(f_{\mathrm{sec}}(C_0))$. Therefore, f is secrecy-preserving for $y \mapsto (y, f_{\mathrm{sec}})$.

Let us prove the second point. Let x_A, x_B, k_A'', k_B'' be fresh variables not in V'. Let $\widetilde{msg}'_A$ (resp. $\widetilde{msg}'_B$) be the sequence of variables corresponding to $\widetilde{msg}_A$ (resp. $\widetilde{msg}_B$), but using variables x_c of the relay process $\mathsf{Relay}(C, \emptyset, \emptyset)$ instead of variables of C. As above, the goal of the relay process $\mathsf{Relay}(C, \emptyset, \emptyset)$ is to capture in variables $\widetilde{msg}'_A$, visible outside C, the content of $\widetilde{msg}_A$, which are variables internal to C but sent or received on public channels, and similarly for $\widetilde{msg}_B$. Since $\mathsf{Relay}(C, \emptyset, \emptyset)$ renames channels $c \in \mathsf{ch}(C)$ to fr_c, we use the relay process $\mathsf{Relay}'(C, \emptyset)$ to perform the reverse renaming.

Let

$$\begin{aligned}C_4 = {}&\mathbf{newChannel}\ \mathsf{ch}'(C); (\mathsf{Relay}'(C, \emptyset) \mid\\ &\quad \mathbf{newChannel}\ c_A, c_B, \mathsf{ch}(C); (\mathsf{Relay}(C, \emptyset, \emptyset) \mid\\ &\quad !^{\widetilde{i} \leq \widetilde{n}} c_A[\widetilde{i}](x_A : T_A); \mathbf{if\ defined}(k'_A[\widetilde{i}], \widetilde{msg}'_A[\widetilde{i}])\ \mathbf{then}\\ &\qquad \mathbf{let}\ k''_A = k'_A[\widetilde{i}]\ \mathbf{in}\ \overline{c'_A[\widetilde{i}]}\langle x_A \rangle; Q'_{2A}\{k''_A/k, \mathsf{sid}(\widetilde{msg}'_A[\widetilde{i}])/x\} \mid\\ &\quad !^{\widetilde{i}' \leq \widetilde{n}'} c_B[\widetilde{i}'](x_B : T_B); \mathbf{if\ defined}(k_B[\widetilde{i}'], \widetilde{msg}'_B[\widetilde{i}'])\ \mathbf{then}\\ &\qquad \mathbf{let}\ k''_B = k_B[\widetilde{i}']\ \mathbf{in}\ \overline{c'_B[\widetilde{i}']}\langle x_B \rangle; Q'_{2B}\{k''_B/k, \mathsf{sid}(\widetilde{msg}'_B[\widetilde{i}'])/x\} \mid [\,]))\end{aligned}$$

By Lemma 4, there exists an evaluation context C_4' such that

$$C''_{\mathsf{h}}[C_4[Q_1]] \approx_0^{V'} C'_4[C_{\mathsf{h}}[Q_1]] = C'_4[S_1]$$

where the context C_4' runs in time at most $t_{C_4} \leq t_2$, calls the l-th hash oracle in C_{h} at most $n_{\mathsf{h},l,C_4} \leq n_{\mathsf{h},l,2}$ times, so $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$, and its other parameters are the same as those of C_4, that is, it does not alter the other parameters.

We have

$$\begin{aligned}C''_{\mathsf{h}}[C_4[Q_1]] \approx_0^{V'} C''_{\mathsf{h}}[C[&\mathbf{event}\ e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \mathbf{let}\ k'_A = k_A\ \mathbf{in\ if\ defined}(k'_A[\widetilde{i}])\ \mathbf{then}\\ &\quad \mathbf{let}\ k''_A = k'_A[\widetilde{i}]\ \mathbf{in}\ \overline{c'_A[\widetilde{i}]}\langle M_A \rangle; (Q_{1A} \mid Q'_{2A}\{k''_A/k, \mathsf{sid}(\widetilde{msg}_A)/x\}),\\ &\mathbf{event}\ e_B(\mathsf{sid}(\widetilde{msg}_B), k_B); \mathbf{if\ defined}(k_B[\widetilde{i}'])\ \mathbf{then}\\ &\quad \mathbf{let}\ k''_B = k_B[\widetilde{i}']\ \mathbf{in}\ \overline{c'_B[\widetilde{i}']}\langle M_B \rangle; (Q_{1B} \mid Q'_{2B}\{k''_B/k, \mathsf{sid}(\widetilde{msg}_B)/x\})]]\end{aligned}$$

by eliminating communications on the private channels c_A, c_B, $\mathsf{ch}(C)$, $\mathsf{ch}'(C)$ (Appendix A.2), so

$$\begin{aligned}C'_4[S_1] &\approx_0^{V'} C''_{\mathsf{h}}[C_4[Q_1]]\\ &\approx_0^{V'} C''_{\mathsf{h}}[C[\mathbf{event}\ e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \overline{c'_A[\widetilde{i}]}\langle M_A \rangle; (Q_{1A} \mid Q'_{2A}\{k_A/k, \mathsf{sid}(\widetilde{msg}_A)/x\}),\\ &\qquad \mathbf{event}\ e_B(\mathsf{sid}(\widetilde{msg}_B), k_B); \overline{c'_B[\widetilde{i}']}\langle M_B \rangle; (Q_{1B} \mid Q'_{2B}\{k_B/k, \mathsf{sid}(\widetilde{msg}_B)/x\})]]\\ &\qquad = S_{composed}\end{aligned}$$

since k'_A is an abbreviation for $k'_A[\widetilde{i}]$, so $k'_A[\widetilde{i}]$ is always defined and $k''_A = k'_A = k_A$, and similarly for k_B. We can remove the assignment to k'_A since k'_A is not used and $k'_A \notin V'$. □

A.10 Variant with Several Holes

In Theorem 2, event e_B appears in a single hole of the context C. Theorem 4 generalizes it to several holes.

Theorem 4 (Variant with several holes for event e_B)**.** *Let C be any context with $J+1$ holes, with replications $!^{\widetilde{i} \le \widetilde{n}}$ above the first hole and $!^{\widetilde{i}'_j \le \widetilde{n}'_j}$ above the $(j+1)$-th hole $(1 \le j \le J)$ and without* **event_abort**. *Let Q_{1A} and $Q_{1B,j}$ $(1 \le j \le J)$ be processes without* **event_abort**. *Let $k, k_A, k_{B,j}$ $(1 \le j \le J)$ be variables of type T. Let*

$$\begin{aligned}
&Q_1 = C[\textbf{event } e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \textbf{let } k'_A = k_A \textbf{ in } \overline{c_A[\widetilde{i}]}\langle M_A\rangle; Q_{1A},\\
&\qquad (\textbf{event } e_B(\mathsf{sid}(\widetilde{msg}_{B,j}), k_{B,j}); \overline{c_{B,j}[\widetilde{i}'_j]}\langle M_{B,j}\rangle; Q_{1B,j})_{1\le j\le J}]\\
&Q_2 = c_1(); \textbf{new } k : T; \overline{c_2}\langle\rangle; (Q_{2A} \mid Q_{2B})\\
&S_1 = C_{\mathsf{h}}[Q_1]\\
&S_2 = C'_{\mathsf{h}}[\mathsf{AddReplSid}(\widetilde{i} \le \widetilde{n}, c'_1, T_{\mathsf{sid}}, Q_2)]
\end{aligned}$$

where Q_1 and Q_2 are hash-well-formed; $\widetilde{msg}_A$ is a sequence of variables defined in C above the first hole and input or output by C above the first hole or by the output $\overline{c_A[\widetilde{i}]}\langle M_A\rangle$; for all $j \in \{1,\dots,J\}$, $\widetilde{msg}_{B,j}$ is a sequence of variables input or output by C above the $(j+1)$-th hole; sid *is a function that takes a sequence of messages and returns a session identifier of type T_{sid}; C, Q_{1A}, $Q_{1B,j}$ $(1 \le j \le J)$, Q_{2A}, and Q_{2B} make all their inputs and outputs on pairwise distinct channels with indices the current replication indices; c_A, $c_{B,j}$ $(1 \le j \le J)$, c_1, c'_1, c_2, k'_A, e_A, e_B do not occur elsewhere in S_1, S_2; S_1 and S_2 have no common variable, no common channel, no common event, and no common table; S_1 and S_2 do not contain* **newChannel**; *and there is no* **defined** *condition in Q_2.*

Let $Q'_{2A} = \mathsf{AddIdxSid}(\widetilde{i} \le \widetilde{n}, x : T_{\mathsf{sid}}, Q_{2A})$ and $Q'_{2B,j} = \mathsf{AddIdxSid}(\widetilde{i}'_j \le \widetilde{n}'_j, x : T_{\mathsf{sid}}, Q_{2B})$. Let $c'_A, c'_{B,j}$ $(1 \le j \le J)$ be fresh channels. Let

$$\begin{aligned}
&Q_{composed} =\\
&\quad C[\textbf{event } e_A(\mathsf{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \overline{c'_A[\widetilde{i}]}\langle M_A\rangle; (Q_{1A} \mid Q'_{2A}\{k_A/k, \mathsf{sid}(\widetilde{msg}_A)/x\}),\\
&\qquad (\textbf{event } e_B(\mathsf{sid}(\widetilde{msg}_{B,j}), k_{B,j}); \overline{c'_{B,j}[\widetilde{i}']}\langle M_{B,j}\rangle; (Q_{1B,j} \mid Q'_{2B,j}\{k_{B,j}/k, \mathsf{sid}(\widetilde{msg}_{B,j})/x\}))_{1\le j\le J}]\\
&S_{composed} = C''_{\mathsf{h}}[Q_{composed}]
\end{aligned}$$

Let $S^{\circ}_{composed}$ be obtained from $S_{composed}$ by removing all events of S_1.

Then we have the same properties as in Theorem 2 with $\sum_{j=1}^{J} \prod \widetilde{n}'_j$ instead of $\prod \widetilde{n}'$ and C'' independent of Q_{1A} and $Q_{1B,j}$ $(1 \le j \le J)$.

Proof. The proof is the same as for Theorem 2. We just need to repeat the treatment of the hole that contains event e_B for each $j \in \{1,\dots,J\}$; the definition of $\mathsf{fresh}(\widetilde{i}'', \widetilde{u})$ is updated into $\mathsf{fresh}(\widetilde{i}'', (\widetilde{u}_j)_{1\le j\le J})$ = **find** $\bigoplus_{j=1}^{J} \widetilde{u}'_j = \widetilde{i}'''_j \le \widetilde{n}'_j$ **suchthat defined**$(\widetilde{u}_j[\widetilde{i}'''_j]) \wedge \widetilde{u}_j[\widetilde{i}'''_j] = \widetilde{i}''$ **then** false **else** true, using a **find** with several branches, so that we have $\mathsf{fresh}(\widetilde{i}'', (\widetilde{u}_j)_{1\le j\le J})$ when $\widetilde{i}''$ was not used before, that is, it does not occur in any array $\widetilde{u}_j$ for $j \in \{1,\dots,J\}$. □

A.11 Proof for Section 5.5

Proof of Theorem 3. The proof of this theorem is very similar to the proof of Theorem 2, so we do not repeat it, but just point out the differences. Let C_5 be defined as in the proof of

Theorem 2 except that the condition $\mathsf{fresh}(\widetilde{i''}, \widetilde{u})$ is removed, because we do not have injectivity, and the process $\mathsf{Relay}(Q_{2B}, \widetilde{i'}, \widetilde{u})$ is replaced with $\mathsf{Relay}(Q_{2B}, \widetilde{i'}, (\widetilde{u}, \widetilde{i'}))$. Indeed, the indices of Q_{2B} are renumbered in a different way: the indices of Q_{2B} in the composed system are $\widetilde{i'}$; in Theorem 2, they are mapped to $\widetilde{u}[\widetilde{i'}]$ in S_2, while in this theorem, they are mapped to $\widetilde{u}[\widetilde{i'}], \widetilde{i'}$ in S_2. This change affects the rest of proof as we detail below. Let G_0 be defined as in the proof of Theorem 2, except that the condition $\mathsf{fresh}(\widetilde{i''}, \widetilde{u})$ is removed. Let Q'_2, Q''_{2B}, and Q''_2 be defined as in the proof of Theorem 2. We have $C''_{\mathsf{h}}[C_5[Q''_2]] \approx_0^{V_1 \cup \{\widetilde{u}\}} C''_{\mathsf{h}}[C_5[Q'_2]]$, as in Theorem 2.

Next, we show $G_0 \overset{\sim}{\to}{}^{V_1, V_1 \cup \{\widetilde{u}\}}_{f',0} C''_{\mathsf{h}}[C_5[Q''_2]]$. To prove this property, we consider an evaluation context C_6 acceptable for G_0 with public variables V_1. Let $f'(C_6) = C'_6$ be obtained from C_6 by replacing array accesses $y[\widetilde{M}, \widetilde{M'}]$ with $y[\widetilde{u}[\widetilde{M}], \widetilde{M}, \widetilde{M'}]$, when $y \in V_1$ is defined in Q'_{2B}, $\widetilde{M}$ contains as many elements as $\widetilde{i'}$, and $\widetilde{M'}$ contains the other indices of y if any. We establish a correspondence between the traces of $C_6[G_0]$ and those of $C'_6[C''_{\mathsf{h}}[C_5[Q''_2]]]$: we eliminate communications on the private channels c'_1, c_2, $\mathsf{ch}(Q_{2B})$, $\mathsf{ch}'(Q_{2B})$ (Appendix A.2) and we renumber the variables of Q'_{2B}, replacing indices $\widetilde{a} \leq \widetilde{n}'$ in $C_6[G_0]$ with $\widetilde{u}[\widetilde{a}], \widetilde{a} \leq \widetilde{n}, \widetilde{n}'$ in $C'_6[C''_{\mathsf{h}}[C_5[Q''_2]]]$. In this correspondence, an execution of $Q'_{2B} = \mathsf{AddIdxSid}(\emptyset \leq \emptyset, x : T_{\mathsf{sid}}, Q_{2B})$ with replication indices $\widetilde{a}$ in $C_6[G_0]$ corresponds to an execution of $Q''_{2B} = \mathsf{AddIdxSid}(\widetilde{i} \leq \widetilde{n}, x : T_{\mathsf{sid}}, Q_{2B})$ with replication indices $\widetilde{u}[\widetilde{a}], \widetilde{a}$ in $C'_6[C''_{\mathsf{h}}[C_5[Q''_2]]]$. The proofs of $G_0 \overset{\sim}{\to}{}^{V_1, V_1 \cup \{\widetilde{u}\}}_{f',0} C''_{\mathsf{h}}[C_5[Q''_2]]$ and of $S^{\circ}_{composed} \overset{\sim}{\to}{}^{V_1, V_2}_{f,p_3} S_2$ then proceed as in the proof of Theorem 2: the removal of the condition fresh compensates for the non-injectivity of the correspondence (7).

For $y \in V_2 \cap \mathsf{var}(Q_{2A})$, f is secrecy-preserving for $y \mapsto (y, f_{\mathrm{sec}})$ where $f_{\mathrm{sec}}(C_3) = f'(C_3)[C'_5[\,]]$, by Lemma 10, as in Theorem 2. For $y \in V_2 \cap \mathsf{var}(Q_{2B})$, the proof needs to be adapted since the renumbering of variables has changed. Suppose that S_2 preserves the secrecy of y with public variables $V_2 \setminus \{y\}$ up to probability p_2. By Lemma 1, $C'_5[S_2]$ preserves the secrecy of y with public variables $(V' \cup \mathsf{var}(C'_5)) \setminus \{y\}$ up to probability p'_2 such that $p'_2(C_5) = p_1(C_5[C'_5])$. Let R_y be the process used for testing secrecy of y in $S^{\circ}_{composed}$. Let R'_y be obtained by renumbering the variables R_y: $R'_y = f'(R_y)$, that is,

$$
\begin{aligned}
R'_y = {}& c_{s0}(); \mathbf{new}\ b : \mathit{bool}; \overline{c_{s0}}\langle\rangle; \\
& (!^{i_s \leq n_s} c_s[i_s](\widetilde{u}_1 \leq \widetilde{n}', \widetilde{u}_2 \leq \widetilde{n}_2); \mathbf{if\ defined}(y[\widetilde{u}[\widetilde{u}_1], \widetilde{u}_1, \widetilde{u}_2])\ \mathbf{then} \\
& \quad \mathbf{if}\ b\ \mathbf{then}\ \overline{c_s[i_s]}\langle y[\widetilde{u}[\widetilde{u}_1], \widetilde{u}_1, \widetilde{u}_2]\rangle\ \mathbf{else} \\
& \quad \mathbf{find}\ u'_s = i'_s \leq n_s\ \mathbf{suchthat\ defined}(y'[i'_s], \widetilde{u}_1[i'_s], \widetilde{u}_2[i'_s]) \wedge \widetilde{u}_1[i'_s] = \widetilde{u}_1 \wedge \widetilde{u}_2[i'_s] = \widetilde{u}_2 \\
& \quad \mathbf{then}\ \overline{c_s[i_s]}\langle y'[u'_s]\rangle\ \mathbf{else\ new}\ y' : T; \overline{c_s[i_s]}\langle y'\rangle \\
& \mid c'_s(b'); \mathbf{if}\ b = b'\ \mathbf{then\ event\_abort}\ \mathsf{S}\ \mathbf{else\ event\_abort}\ \overline{\mathsf{S}})
\end{aligned}
$$

Let c''_s be a fresh channel, $\widetilde{u}'_1$, $\widetilde{u}'_2$, y'', $\widetilde{u}''_1$, $\widetilde{u}''_2$, y''' be fresh variables, and let us define relay processes:

$$
\begin{aligned}
\mathsf{Relay}_R = {}& !^{i_s \leq n_s} c''_s[i_s](\widetilde{u}'_1 \leq \widetilde{n}', \widetilde{u}'_2 \leq \widetilde{n}_2); \mathbf{if\ defined}(\widetilde{u}[\widetilde{u}'_1])\ \mathbf{then} \\
& \overline{c_s[i_s]}\langle \widetilde{u}[\widetilde{u}'_1], \widetilde{u}'_1, \widetilde{u}'_2\rangle; c_s[i_s](y'' : T); \overline{c''_s[i_s]}\langle y''\rangle \\
\mathsf{Relay}'_R = {}& !^{i_s \leq n_s} c_s[i_s](\widetilde{u}''_1 \leq \widetilde{n}', \widetilde{u}''_2 \leq \widetilde{n}_2); \overline{c''_s[i_s]}\langle \widetilde{u}''_1, \widetilde{u}''_2\rangle; c''_s[i_s](y''' : T); \overline{c_s[i_s]}\langle y'''\rangle
\end{aligned}
$$

The process Relay_R renumbers the queries for y like f' renumbers y. However, it uses channel c''_s instead of c_s, so we use process Relay'_R to revert to channel c_s. Let $C_R = \mathbf{newChannel}\ c''_s$; $(\mathsf{Relay}'_R \mid \mathbf{newChannel}\ c_s; (\mathsf{Relay}_R \mid [\,]))$. Let R''_y be the process used for testing secrecy of y in $C'_5[S_2]$. (The process R''_y differs from R_y because y has indices $\widetilde{i'} \leq \widetilde{n}', \widetilde{i}_2 \leq \widetilde{n}_2$ in $S^{\circ}_{composed}$, while it has indices $\widetilde{i} \leq \widetilde{n}, \widetilde{i'} \leq \widetilde{n}', \widetilde{i_2} \leq \widetilde{n}_2$ in $C'_5[S_2]$.) We have

$$C'_5[S_2] \mid R'_y \approx_0^{V_1} C_R[C'_5[S_2] \mid R''_y]$$

by eliminating communications on the private channels c_s and c''_s (Appendix A.2). The equality test $\widetilde{u}_1[i'_s] = \widetilde{u}_1$ performed by R'_y in the left-hand side becomes $\widetilde{u}[\widetilde{u}_1[i'_s]] = \widetilde{u}[\widetilde{u}_1] \wedge \widetilde{u}_1[i'_s] = \widetilde{u}_1$ in the right-hand side because R''_y always receives $\widetilde{u}[\widetilde{u}_1], \widetilde{u}_1$ instead of $\widetilde{u}_1$. It is easy to see that these two tests are equivalent: the second test includes the first one as a conjunct, and conversely when $\widetilde{u}_1[i'_s] = \widetilde{u}_1$ holds, we obviously have $\widetilde{u}[\widetilde{u}_1[i'_s]] = \widetilde{u}[\widetilde{u}_1]$. The rest of the proof that f is secrecy-preserving for $y \mapsto (y, f_{\mathrm{sec}})$ proceeds exactly as in Theorem 2.

The proof of the second point of the theorem also proceeds as in Theorem 2. □

A.12 Single Process with Key Reuse

The next theorem is a weakened variant of Theorem 1 that allows us to reuse the same key k several times. This situation complicates the theorem considerably.

Theorem 5. *Let C be any context with one hole, with replications $!^{\widetilde{i} \leq \widetilde{n}}$ above the hole and without* **event_abort**. *Let Q_1 be a process without* **event_abort**. *Let M be a term of type T. Let*

$$\begin{aligned}
Q'_1 = C[&\textbf{let } k = M \textbf{ in event } e(\mathsf{sid}(\widetilde{msg}), k);\\
&\textbf{find } \widetilde{u} = \widetilde{i'} \leq \widetilde{n} \textbf{ suchthat defined}(\widetilde{msg}[\widetilde{i'}], k_1[\widetilde{i'}]) \wedge \mathsf{sid}(\widetilde{msg}[\widetilde{i'}]) = \mathsf{sid}(\widetilde{msg})\\
&\textbf{then } \overline{c_1[\widetilde{i}]}\langle M_1\rangle; Q_1\\
&\textbf{else let } k_1 = k \textbf{ in } \overline{c'_1[\widetilde{i}]}\langle M_1\rangle; Q_1]\\
Q_2 = \; & !^{\widetilde{i''} \leq \widetilde{n}} Q_0\\
Q'_2 = \; & c_2(); \textbf{new } k : T; \overline{c_3}\langle\rangle; Q_2\\
S_1 = \; & C_{\mathsf{h}}[Q'_1]\\
S_2 = \; & C'_{\mathsf{h}}[\mathsf{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c'_2, T_{\mathsf{sid}}, Q'_2)]
\end{aligned}$$

where Q'_1 and Q'_2 are hash-well-formed; $\widetilde{msg}$ is a sequence of variables input or output by C above the hole; sid *is a function that takes a sequence of messages and returns a session identifier of type T_{sid}; c_1, c'_1, c_2, c'_2, c_3, e, k_1, $\widetilde{u}$ do not occur elsewhere in S_1, S_2; k does not occur elsewhere in S_1 (k occurs in Q_0); k is the only common variable between S_1 and S_2; S_1 and S_2 have no common channel, no common event, and no common table; S_1 and S_2 do not contain* **newChannel**; *Q_2 contains no* **defined** *condition; and C and Q_2 make all their inputs and outputs on pairwise distinct channels with indices the current replication indices. Let $\mathsf{ch}(Q_0)$ be the channels of Q_0. For each channel c in $\mathsf{ch}(Q_0)$, let fr_c be a fresh channel.*

Let $Q'_0 = \mathsf{AddIdxSid}(\emptyset \leq \emptyset, x : T_{\mathsf{sid}}, Q_0)$. Let c''_1 be a fresh channel. Let

$$\begin{aligned}
Q_{composed} &= C[\textbf{let } k = M \textbf{ in event } e(\mathsf{sid}(\widetilde{msg}), k); \overline{c''_1[\widetilde{i}]}\langle M_1\rangle;\\
&\qquad (Q_1 \mid Q'_0\{\widetilde{i}/\widetilde{i''}, \mathsf{sid}(\widetilde{msg})/x, \mathsf{fr}_c/c, c \in \mathsf{ch}(Q_0)\})]\\
S_{composed} &= C''_{\mathsf{h}}[Q_{composed}]
\end{aligned}$$

Let $S^\circ_{composed}$ be obtained from $S_{composed}$ by removing the events of S_1.

Let $t_1 = t_C + \prod \widetilde{n} \times (t_M + t_{Q_1})$ be an upper bound on the runtime of Q'_1, $t_2 = \prod \widetilde{n} \times t_{Q_0}$ be an upper bound on the runtime of Q'_0 in $Q_{composed}$, $n_{\mathsf{h},l,1} = n_{\mathsf{h},l,C} + \prod \widetilde{n} \times (n_{\mathsf{h},l,M} + n_{\mathsf{h},l,Q_1})$, and $n_{\mathsf{h},l,2} = \prod \widetilde{n} \times n_{\mathsf{h},l,Q_0}$.

1. *If S_1 preserves the secrecy of k_1 with public variables V ($V \subseteq \mathsf{var}(S_1) \setminus (\{k_1\} \cup \mathsf{var}(C_\mathsf{h}))$) up to probability p and S_1 satisfies the correspondence* $\mathbf{event}(e(sid, k)) \wedge \mathbf{event}(e(sid, k')) \Longrightarrow k = k'$ *with public variables $V \cup \{k_1\}$ up to probability p', then there exists an evaluation context $C_5'^\circ$ such that, for any $V_1 \subseteq V \cup (\mathsf{var}(Q_2') \setminus (\{k\} \cup \mathsf{var}(C'_\mathsf{h})))$, we have $S^\circ_{composed} \approx^{V_1}_{p''} C_5'^\circ[S_2]$, $C_5'^\circ$ is acceptable for S_2 without public variables and contains no event, and $C_5'^\circ$ runs in time at most t_1, calls the l-th hash oracle at most $n_{\mathsf{h},l,1}$ times, so $n'_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,1}$, and does not alter the other parameters, where $p''(C_3, t_D) = p(C_3' + t_D) + p'(C_3', t_D)$ and, assuming C_3 calls the l-th hash oracle at most $n''_{\mathsf{h},l}$ times, the context C_3' runs in time at most $t_{C_3} + t_2$, calls the l-th hash oracle at most $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$ times, and its other parameters are the same as those of C_3.*
2. *There exists an evaluation context C_4' such that, for any $V' \subseteq \mathsf{var}(S_{composed}) \setminus (\{k_1, \widetilde{u}\} \cup \mathsf{var}(C''_\mathsf{h}))$, we have $S_{composed} \approx^{V'}_0 C_4'[S_1]$ and C_4' is acceptable for S_1 with public variable k, contains the events of S_2, runs in time at most t_2, calls the l-th hash oracle at most $n_{\mathsf{h},l,2}$ times, so $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$, and does not alter the other parameters.*

Moreover, $C_5'^\circ$ is independent of the details of Q_0: it depends only on the channels of Q_0, whether they are for input or for output, under which replications and with which type of data; C_4' is independent of Q_1.

In this theorem, the system S_1 establishes a key k and the system S_2 creates a fresh key k and runs several sessions of Q_0 using this key. The composed system runs S_1 and Q_0 using the key k provided by S_1. As in Theorem 2, these systems can share hash oracles, included in C_h, C'_h, and C''_h.

Inside S_1, the **find** and the assignment $k_1 = k$ in Q_1' just serve for specifying the desired security properties of S_1, as follows. When the session identifier $\mathsf{sid}(\widetilde{msg})$ has not been seen before, S_1 stores its key k in k_1, and we require that k_1 be secret. As a consequence of secrecy, the keys obtained with different session identifiers are indistinguishable from independent random numbers. Furthermore, the correspondence $\mathbf{event}(e(sid, k)) \wedge \mathbf{event}(e(sid, k')) \Longrightarrow k = k'$ guarantees that, when another session has the same session identifier sid, then it also has the same key. Since the session identifiers can be computed from public messages $\widetilde{msg}$, the adversary knows whether S_1 reuses the same key or uses a fresh key. Therefore, a context around S_1 can know whether it should run another session of Q_0 with the same key or a new session of Q_0 with a fresh key. This observation is important for the proof of the first point of Theorem 5.

As in the previous theorems, the first point of Theorem 5 allows us to transfer security properties from S_2 to the composed system $S_{composed}$, and the second point allows us to transfer security properties from S_1 to $S_{composed}$.

Proof of Theorem 5. We use the notations of relay processes of the proof of Theorem 2. Let k' and r be fresh variables. Let

$$
\begin{aligned}
C_5 = \ & \mathbf{newChannel}\ c_2', c_3, \mathsf{ch}(Q_0);\\
& (C[\mathbf{let}\ k' = M\ \mathbf{in\ event}\ e(\mathsf{sid}(\widetilde{msg}), k');\\
& \quad \mathbf{find}\ \widetilde{u} = \widetilde{i'} \le \widetilde{n}\ \mathbf{suchthat\ defined}(r[\widetilde{i'}], \widetilde{msg}[\widetilde{i'}]) \wedge \mathsf{sid}(\widetilde{msg}[\widetilde{i'}]) = \mathsf{sid}(\widetilde{msg})\\
& \quad \mathbf{then}\ \overline{c_1''[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid \mathsf{Relay}(Q_0, \widetilde{i}, (\widetilde{u}, \widetilde{i})))\\
& \quad \mathbf{else\ let}\ r = \mathsf{cst}\ \mathbf{in}\ \overline{c_2'[\widetilde{i}]}\langle \mathsf{sid}(\widetilde{msg})\rangle; c_3[\widetilde{i}](); \overline{c_1''[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid \mathsf{Relay}(Q_0, \widetilde{i}, (\widetilde{i}, \widetilde{i})))]\\
& \mid [\,])
\end{aligned}
$$

In the **then** branch, the key k is the same as for indices $\widetilde{u}$, so we reuse it. In the **else** branch, a new key k is created by sending a message on $c_2'[\widetilde{i}]$ and receiving the reply on $c_3[\widetilde{i}]$.

Let $Q_2'' = \mathsf{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c_2', T_{\mathsf{sid}}, Q_2')$, so that we have $S_2 = C_{\mathsf{h}}'[Q_2'']$. By Lemma 4, there exists an evaluation context C_5' such that

$$C_{\mathsf{h}}''[C_5[Q_2'']] \approx_0^{V_1} C_5'[C_{\mathsf{h}}'[Q_2'']] = C_5'[S_2] \tag{21}$$

where the context C_5' runs in time at most $t_{C_5} \leq t_1$, calls the l-th hash oracle in C_{h}' at most $n_{\mathsf{h},l,C_5} \leq n_{\mathsf{h},l,1}$ times, so $n'_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,1}$, and its other parameters are the same as those of C_5.

Let

$$\begin{aligned}
G_1 = C_{\mathsf{h}}''[&\textbf{newChannel}\ \mathsf{ch}(Q_0); C[\\
&\quad \textbf{let}\ k' = M\ \textbf{in event}\ e(\mathsf{sid}(\widetilde{msg}), k');\\
&\quad \textbf{find}\ \widetilde{u} = \widetilde{i}' \leq \widetilde{n}\ \textbf{suchthat defined}(r[\widetilde{i}'], \widetilde{msg}[\widetilde{i}']) \wedge \mathsf{sid}(\widetilde{msg}[\widetilde{i}']) = \mathsf{sid}(\widetilde{msg})\\
&\quad \textbf{then}\ \overline{c_1''[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid \mathsf{Relay}(Q_0, \widetilde{i}, (\widetilde{u}, \widetilde{i})))\\
&\quad \textbf{else}\\
&\quad\quad \textbf{let}\ r = \mathsf{cst}\ \textbf{in new}\ k : T; \overline{c_1''[\widetilde{i}]}\langle M_1\rangle;\\
&\quad\quad (Q_1 \mid \mathsf{Relay}(Q_0, \widetilde{i}, (\widetilde{i}, \widetilde{i})) \mid \mathsf{AddIdxSid}(\widetilde{i} \leq \widetilde{n}, x : T_{\mathsf{sid}}, Q_2)\{\mathsf{sid}(\widetilde{msg})/x\})]]
\end{aligned}$$

We have $C_{\mathsf{h}}''[C_5[Q_2'']] \approx_0^{V_1} G_1$ by eliminating communications on c_2' and c_3 (Appendix A.2). Note that the **find** introduced at the root of Q_2'' by $\mathsf{AddReplSid}$ never succeeds because the session identifier $\mathsf{sid}(\widetilde{msg})$ is sent on c_2' only when it was not seen before. Let

$$\begin{aligned}
G_2 = C_{\mathsf{h}}''[&\textbf{newChannel}\ \mathsf{ch}(Q_0); C[\\
&\quad \textbf{let}\ k = M\ \textbf{in event}\ e(\mathsf{sid}(\widetilde{msg}), k);\\
&\quad \textbf{find}\ \widetilde{u} = \widetilde{i}' \leq \widetilde{n}\ \textbf{suchthat defined}(k_1[\widetilde{i}'], \widetilde{msg}[\widetilde{i}']) \wedge \mathsf{sid}(\widetilde{msg}[\widetilde{i}']) = \mathsf{sid}(\widetilde{msg})\\
&\quad \textbf{then}\ \overline{c_1''[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid \mathsf{Relay}(Q_0, \widetilde{i}, (\widetilde{u}, \widetilde{i})))\\
&\quad \textbf{else}\\
&\quad\quad \textbf{new}\ k_1 : T; \overline{c_1''[\widetilde{i}]}\langle M_1\rangle;\\
&\quad\quad (Q_1 \mid \mathsf{Relay}(Q_0, \widetilde{i}, (\widetilde{i}, \widetilde{i})) \mid \mathsf{AddIdxSid}(\widetilde{i} \leq \widetilde{n}, x : T_{\mathsf{sid}}, Q_2)\{k_1/k, \mathsf{sid}(\widetilde{msg})/x\})]]
\end{aligned}$$

We also have $G_1 \approx_0^{V_1} G_2$ by renaming k into k_1 and k' into k, and using the definition of $k_1[\widetilde{i}']$ instead of the definition of $r[\widetilde{i}']$, which is equivalent since these variables are defined at the same time. That allows us to remove the variable r. Therefore, by transitivity, $C_5'[S_2] \approx_0^{V_1} G_2$. Let $C_5'^{\circ}$ and G_2° be obtained from C_5' and G_2 respectively by removing the events of S_1. By Lemma 11, we have

$$C_5'^{\circ}[S_2] \approx_0^{V_1} G_2^{\circ} \tag{22}$$

Let

$$\begin{aligned}
Q_1'' = C[&\textbf{let}\ k = M\ \textbf{in event}\ e(\mathsf{sid}(\widetilde{msg}), k);\\
&\textbf{find}\ \widetilde{u} = \widetilde{i}' \leq \widetilde{n}\ \textbf{suchthat defined}(k_1[\widetilde{i}'], \widetilde{msg}[\widetilde{i}']) \wedge \mathsf{sid}(\widetilde{msg}[\widetilde{i}']) = \mathsf{sid}(\widetilde{msg})\\
&\textbf{then}\ \overline{c_1[\widetilde{i}]}\langle M_1\rangle; Q_1\\
&\textbf{else new}\ k_1 : T; \overline{c_1'[\widetilde{i}]}\langle (k_1, M_1)\rangle; Q_1]
\end{aligned}$$

Let G_3 be obtained by replacing **new** $k_1 : T$ with **let** $k_1 = k$ **in** in G_2. Let Q_1''' be obtained by replacing **new** $k_1 : T$ with **let** $k_1 = k$ **in** in Q_1''. Let G_3°, $Q_1''^{\circ}$, and $Q_1'''^{\circ}$ be obtained from G_3, Q_1'', and Q_1''' respectively by removing the events of S_1.

By introducing a relay process and renaming channels as in the proof of Theorem 2, we have

$$G_2^\circ\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C)\} \approx_0^{V_1} C''_{\mathsf{h}}[C_1[Q''^{\circ}_1]] \tag{23}$$

$$G_3^\circ\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C)\} \approx_0^{V_1} C''_{\mathsf{h}}[C_1[Q'''^{\circ}_1]] \tag{24}$$

for some evaluation context C_1 that runs in time at most t_2. By Lemma 4, there exists an evaluation context C'_1 such that

$$C''_{\mathsf{h}}[C_1[Q''^{\circ}_1]] \approx_0^{V_1} C'_1[C_{\mathsf{h}}[Q''^{\circ}_1]] \tag{25}$$

$$C''_{\mathsf{h}}[C_1[Q'''^{\circ}_1]] \approx_0^{V_1} C'_1[C_{\mathsf{h}}[Q'''^{\circ}_1]] \tag{26}$$

where the context C'_1 runs in time at most $t_{C_1} \leq t_2$, calls the l-th hash oracle in C_{h} at most $n_{\mathsf{h},l,C_1} \leq n_{\mathsf{h},l,2}$ times, so $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$, and its other parameters are the same as those of C_1.

Since $S_1 = C_{\mathsf{h}}[Q'_1]$ preserves the secrecy of k_1 with public variables V ($k_1, k \notin V$) up to probability p, by Lemma 9, we have $C_{\mathsf{h}}[Q''^{\circ}_1] \approx_{p_0}^{V} C_{\mathsf{h}}[Q'''^{\circ}_1]$ where $p_0(C_3, t_D) = p(C_3 + t_D)$. Therefore,

$$C'_1[C_{\mathsf{h}}[Q''^{\circ}_1]] \approx_{p_1}^{V_1} C'_1[C_{\mathsf{h}}[Q'''^{\circ}_1]] \tag{27}$$

where $p_1(C_3, t_D) = p_0(C_3[C'_1], t_D) = p(C_3[C'_1] + t_D) = p(C'_3 + t_D)$ and the context C'_3 runs in time at most $t_{C_3} + t_2$, calls the l-th hash oracle at most $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$ times, and its other parameters are the same as those of C_3. So by combining (23), (25), (27), (26), and (24),

$$G_2^\circ\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C)\} \approx_{p_1}^{V_1} G_3^\circ\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C)\}$$

so by renaming channels

$$G_2^\circ \approx_{p_1}^{V_1} G_3^\circ \tag{28}$$

Let

$$\begin{aligned}
G_4 = C''_{\mathsf{h}}[C[&\textbf{let } k = M \textbf{ in event } e(\mathsf{sid}(\widetilde{msg}), k);\\
&\textbf{find } \widetilde{u} = \widetilde{i}' \leq \widetilde{n} \textbf{ suchthat defined}(k_1[\widetilde{i}'], \widetilde{msg}[\widetilde{i}']) \wedge \mathsf{sid}(\widetilde{msg}[\widetilde{i}']) = \mathsf{sid}(\widetilde{msg})\\
&\textbf{then } ^{(*)}\ \overline{c''_1[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid Q'_0\{k_1[\widetilde{u}]/k, \widetilde{i}/\widetilde{i}'', \mathsf{sid}(\widetilde{msg})/x, \mathsf{fr}_c/c, c \in \mathsf{ch}(Q_0)\})\\
&\textbf{else let } k_1 = k \textbf{ in } \overline{c''_1[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid Q'_0\{k_1/k, \widetilde{i}/\widetilde{i}'', \mathsf{sid}(\widetilde{msg})/x, \mathsf{fr}_c/c, c \in \mathsf{ch}(Q_0)\})]]
\end{aligned}$$

We have

$$G_3 \approx_0^{V_1} G_4$$

by eliminating communications on $\mathsf{ch}(Q_0)$ (Appendix A.2) and mapping all variables $x[\widetilde{b}, \widetilde{a}, \dots]$ ($\widetilde{b} \leq \widetilde{n}$, $\widetilde{a} \leq \widetilde{n}$) of $\mathsf{AddIdxSid}(\widetilde{i} \leq \widetilde{n}, x : T_{\mathsf{sid}}, Q_2)\{k_1/k, \mathsf{sid}(\widetilde{msg})/x\}$ to $x[\widetilde{a}, \dots]$, which is possible without clashes since there is a single value of $\widetilde{b}$ actually used for each $\widetilde{a}$ ($\widetilde{b} = \widetilde{u}[\widetilde{a}]$ when $\widetilde{u}[\widetilde{a}]$ is defined; $\widetilde{b} = \widetilde{a}$ otherwise). The table entries and the events are the same on both sides because they contain the session identifier $x = \mathsf{sid}(\widetilde{msg})$, not the replication indices $\widetilde{i}$. By introducing again a relay process and renaming channels as in the proof of Theorem 2, we have $G_4\{\mathsf{fr}_c/c, c \in \mathsf{ch}(C)\} \approx_0^{V_1} C''_{\mathsf{h}}[C_2[Q'_1]]$ for some evaluation context C_2 that runs in time at most t_2 and uses the public variable k_1. By Lemma 4, there exists an evaluation context C'_2 such that $C''_{\mathsf{h}}[C_2[Q'_1]] \approx_0^{V_1} C'_2[C_{\mathsf{h}}[Q'_1]] = C'_2[S_1]$ where the context C'_2 runs in time at most $t_{C_2} \leq t_2$, calls the l-th hash oracle in C_{h} at most $n_{\mathsf{h},l,C_2} \leq n_{\mathsf{h},l,2}$ times, so $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$, and its other parameters are the same as those of C_2.

Since S_1 satisfies the correspondence $\textbf{event}(e(sid, k)) \wedge \textbf{event}(e(sid, k')) \implies k = k'$ with public variables $V \cup \{k_1\}$ up to probability p', by Lemma 1, $C'_2[S_1]$ satisfies this correspondence with public variables V_1 up to probability $p'_1(C_3, t_D) = p'(C_3[C'_2], t_D) = p'(C'_3, t_D)$ where the context C'_3 runs in time at most $t_{C_3} + t_2$ and its other parameters are the same as those of C_3, and so do $C''_{\mathsf{h}}[C_2[Q'_1]]$ and G_4. Therefore, we infer that, at the program point $(*)$ in G_4, $k_1[\widetilde{u}] = k[\widetilde{u}] = k$ except in cases of probability p'_1, since $\mathsf{sid}(\widetilde{msg}[\widetilde{u}]) = \mathsf{sid}(\widetilde{msg})$. So we have $G_4 \approx^{V_1}_{p'_1} G_5$ where

$$\begin{aligned}
G_5 = C''_{\mathsf{h}}[C[&\textbf{let } k = M \textbf{ in event } e(\mathsf{sid}(\widetilde{msg}), k);\\
&\textbf{find } \widetilde{u} = \widetilde{i}' \leq \widetilde{n} \textbf{ suchthat defined}(k_1[\widetilde{i}'], \widetilde{msg}[\widetilde{i}']) \wedge \mathsf{sid}(\widetilde{msg}[\widetilde{i}']) = \mathsf{sid}(\widetilde{msg})\\
&\textbf{then } \overline{c''_1[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid Q'_0\{\mathsf{sid}(\widetilde{msg})/x, \widetilde{i}/\widetilde{i}'', \mathsf{fr}_c/c, c \in \mathsf{ch}(Q_0)\})\\
&\textbf{else let } k_1 = k \textbf{ in } \overline{c''_1[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid Q'_0\{\mathsf{sid}(\widetilde{msg})/x, \widetilde{i}/\widetilde{i}'', \mathsf{fr}_c/c, c \in \mathsf{ch}(Q_0)\})]]
\end{aligned}$$

Moreover, $G_5 \approx^{V_1}_0 S_{composed}$ since k_1 is used only in the test of **find** ($k_1 \notin V$) and both branches of **find** execute the same code except for the assignment to k_1, so the **find** and the assignment to k_1 can be removed. Therefore, by transitivity, $G_3 \approx^{V_1}_{p'_1} S_{composed}$, so by Lemma 11,

$$G_3^\circ \approx^{V_1}_{p'_1} S^\circ_{composed} \tag{29}$$

By combining (22), (28), and (29) by transitivity, we obtain

$$C'^{\circ}_5[S_2] \approx^{V_1}_{p_1 + p'_1} S^\circ_{composed}$$

Let us prove the second point. Let T_1 be the type of M_1. Let k' be a fresh variable not in V'. Let $\widetilde{msg}'$ be the sequence of variables corresponding to $\widetilde{msg}$, but using variables of the relay process $\mathsf{Relay}(C, \emptyset, \emptyset)$ instead of variables of C. As above, the goal of the relay process $\mathsf{Relay}(C, \emptyset, \emptyset)$ is to capture in variables $\widetilde{msg}'$, visible outside C, the content of $\widetilde{msg}$, which are variables internal to C but sent or received on public channels. Since $\mathsf{Relay}(C, \emptyset, \emptyset)$ renames channels $c \in \mathsf{ch}(C)$ to fr_c, we use the relay process $\mathsf{Relay}'(C, \emptyset)$ to perform the reverse renaming.

Let

$$\begin{aligned}
C_4 = {}&\textbf{newChannel } \mathsf{ch}'(C); (\mathsf{Relay}'(C, \emptyset) \mid\\
&\textbf{newChannel } c_1, c'_1, \mathsf{ch}(C); (\mathsf{Relay}(C, \emptyset, \emptyset) \mid\\
&!^{\widetilde{i} \leq \widetilde{n}} c_1[\widetilde{i}](y_1 : T_1); \textbf{if defined}(k[\widetilde{i}], \widetilde{msg}'[\widetilde{i}]) \textbf{ then}\\
&\quad \textbf{let } k' = k[\widetilde{i}] \textbf{ in } \overline{c''_1[\widetilde{i}]}\langle y_1\rangle; Q'_0\{k'/k, \widetilde{i}/\widetilde{i}'', \mathsf{sid}(\widetilde{msg}'[\widetilde{i}])/x, \mathsf{fr}_c/c, c \in \mathsf{ch}(Q_0)\} \mid\\
&!^{\widetilde{i} \leq \widetilde{n}} c'_1[\widetilde{i}](y_2 : T_1); \textbf{if defined}(k[\widetilde{i}], \widetilde{msg}'[\widetilde{i}]) \textbf{ then}\\
&\quad \textbf{let } k' = k[\widetilde{i}] \textbf{ in } \overline{c''_1[\widetilde{i}]}\langle y_2\rangle; Q'_0\{k'/k, \widetilde{i}/\widetilde{i}'', \mathsf{sid}(\widetilde{msg}'[\widetilde{i}])/x, \mathsf{fr}_c/c, c \in \mathsf{ch}(Q_0)\} \mid\\
&[\,]))
\end{aligned}$$

By Lemma 4, there exists an evaluation context C'_4 such that

$$C''_{\mathsf{h}}[C_4[Q'_1]] \approx^{V'}_0 C'_4[C_{\mathsf{h}}[Q'_1]] = C'_4[S_1]$$

where the context C'_4 runs in time at most $t_{C_4} \leq t_2$, calls the l-th hash oracle in C_{h} at most $n_{\mathsf{h},l,C_4} \leq n_{\mathsf{h},l,2}$ times, so $n_{\mathsf{h},l} = n''_{\mathsf{h},l} + n_{\mathsf{h},l,2}$, and its other parameters are the same as those of C_4, that is, it does not alter the other parameters.

Hence, we have

$$
\begin{aligned}
&C'_4[S_1] \approx_0^{V'}\\
&\ C''_{\mathsf{h}}[C[\mathbf{let}\ k = M\ \mathbf{in}\ \mathbf{event}\ e(\mathsf{sid}(\widetilde{msg}), k);\\
&\qquad \mathbf{find}\ \widetilde{u} = \widetilde{i'} \leq \widetilde{n}\ \mathbf{suchthatdefined}(\widetilde{msg}[\widetilde{i'}], k_1[\widetilde{i'}]) \wedge \mathsf{sid}(\widetilde{msg}[\widetilde{i'}]) = \mathsf{sid}(\widetilde{msg})\\
&\qquad \mathbf{then}\\
&\qquad\quad (\mathbf{if}\ \mathbf{defined}(k[\widetilde{i}])\ \mathbf{then}\ \mathbf{let}\ k' = k[\widetilde{i}]\ \mathbf{in}\\
&\qquad\quad \overline{c''_1[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid Q'_0\{k'/k, \widetilde{i}/\widetilde{i}'', \mathsf{sid}(\widetilde{msg})/x, \mathsf{fr}_c/c, c \in \mathsf{ch}(Q_0)\}))\\
&\qquad \mathbf{else}\\
&\qquad\quad \mathbf{let}\ k_1 = k\ \mathbf{in}\ \mathbf{if}\ \mathbf{defined}(k[\widetilde{i}])\ \mathbf{then}\\
&\qquad\quad \mathbf{let}\ k' = k[\widetilde{i}]\ \mathbf{in}\ \overline{c''_1[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid Q'_0\{k'/k, \widetilde{i}/\widetilde{i}'', \mathsf{sid}(\widetilde{msg})/x, \mathsf{fr}_c/c, c \in \mathsf{ch}(Q_0)\})]]
\end{aligned}
$$

by eliminating communications on the private channels $\mathsf{ch}'(C)$, c_1, c'_1, $\mathsf{ch}(C)$ (Appendix A.2), so

$$
\begin{aligned}
C'_4[S_1] \approx_0^{V'} C''_{\mathsf{h}}[C[&\mathbf{let}\ k = M\ \mathbf{in}\ \mathbf{event}\ e(\mathsf{sid}(\widetilde{msg}), k);\\
&\quad \mathbf{find}\ \widetilde{u} = \widetilde{i'} \leq \widetilde{n}\ \mathbf{suchthat}\ \mathbf{defined}(\widetilde{msg}[\widetilde{i'}], k_1[\widetilde{i'}]) \wedge \mathsf{sid}(\widetilde{msg}[\widetilde{i'}]) = \mathsf{sid}(\widetilde{msg})\\
&\quad \mathbf{then}\ \overline{c''_1[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid Q'_0\{\widetilde{i}/\widetilde{i}'', \mathsf{sid}(\widetilde{msg})/x, \mathsf{fr}_c/c, c \in \mathsf{ch}(Q_0)\})\\
&\quad \mathbf{else}\ \mathbf{let}\ k_1 = k\ \mathbf{in}\ \overline{c''_1[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid Q'_0\{\widetilde{i}/\widetilde{i}'', \mathsf{sid}(\widetilde{msg})/x, \mathsf{fr}_c/c, c \in \mathsf{ch}(Q_0)\})]]
\end{aligned}
$$

since k is an abbreviation for $k[\widetilde{i}]$, so $k[\widetilde{i}]$ is always defined and $k' = k$. Hence

$$
\begin{aligned}
C'_4[S_1] &\approx_0^{V'} C''_{\mathsf{h}}[C[\mathbf{let}\ k = M\ \mathbf{in}\ \mathbf{event}\ e(\mathsf{sid}(\widetilde{msg}), k);\\
&\qquad\qquad \overline{c''_1[\widetilde{i}]}\langle M_1\rangle; (Q_1 \mid Q'_0\{\widetilde{i}/\widetilde{i}'', \mathsf{sid}(\widetilde{msg})/x, \mathsf{fr}_c/c, c \in \mathsf{ch}(Q_0)\})]]\\
&= S_{composed}
\end{aligned}
$$

since both branches of **find** execute the same code except for the assignment to k_1 and k_1 and $\widetilde{u}$ are not used ($\{k_1, \widetilde{u}\} \cap V' = \emptyset$). □

A.13 Lemma on a Process that Chooses a Bit

In this section, we consider processes P_0 and P_1 that do not contain b_0 and a process

$$Q = c(x : T); \mathbf{new}\ b_0 : \mathit{bool}; \mathbf{if}\ b_0\ \mathbf{then}\ P_1\ \mathbf{else}\ P_0$$

that chooses a random bit b_0 and runs P_0 or P_1 depending on the value of this bit. For $v \in \{0,1\}$, we define $Q_v = c(x : T); P_v$. The process Q_v runs as Q when $b_0 = v$. (0 stands for false and 1 for true.) Let Q_v° be obtained from Q_v by removing all events. We have the following lemma.

Lemma 12. *If Q preserves the secrecy of b_0 with public variables V up to probability p and Q does not contain* **event_abort***, then $Q_0^\circ \approx_{p'}^V Q_1^\circ$ where $p'(C, t_D) = 4p(C + t_D)$.*

Intuitively, the adversary can distinguish Q_0° from Q_1° if and only if it can determine the value of b_0, that is, b_0 is not secret.

Proof. Let C be any acceptable evaluation context for Q_0 and Q_1 with public variables V, and D a distinguisher. Let c_{s0}, c_s, and c'_s be channels that C does not use.

Let C' be a context that runs C but stores events executed by C in its internal state instead of actually executing events, computes D on the stored sequence of events executed by C and stores the result in b'_0, outputs on channel c_{s0} and inputs on channel c_{s0}, outputs on channel c_s, receives the answer b''_0 on channel c_s. If $b'_0 = b''_0$, it chooses a random b' and sends it on channel c'_s. If $b'_0 \neq b''_0$, it sends false on channel c'_s. Such a context C' exists because it can be encoded as a probabilistic Turing machine adversary, which can itself be encoded as a context in CryptoVerif [14, Section 2.8].

When $b_0 = v$, $C'[Q \mid R_x]$ stores in b'_0 the result of $C[Q^\circ_v] : D$. When $b'_0 = v$,

- if $b = \mathsf{true}$ (probability $1/2$), then $b''_0 = b_0 = v$, so b' is random: $b' = \mathsf{true}$ and S is executed with probability $1/4$ and $b' = \mathsf{false}$ and $\overline{\mathsf{S}}$ is executed with probability $1/4$;
- if $b = \mathsf{false}$ (probability $1/2$), then b''_0 is random, so
  - $b''_0 = v$ with probability $1/4$, so b' is random: $b' = \mathsf{true}$ and $\overline{\mathsf{S}}$ is executed with probability $1/8$ and $b' = \mathsf{false}$ and S is executed with probability $1/8$;
  - $b''_0 \neq v$ with probability $1/4$, and in this case $b' = \mathsf{false}$ and S is executed.

When $b'_0 \neq v$,

- if $b = \mathsf{true}$ (probability $1/2$), then $b''_0 = b_0 = v$, so $b' = \mathsf{false}$ and $\overline{\mathsf{S}}$ is executed with probability $1/2$;
- if $b = \mathsf{false}$ (probability $1/2$), then b''_0 is random, so
  - $b''_0 \neq v$ with probability $1/4$, so b' is random: $b' = \mathsf{true}$ and $\overline{\mathsf{S}}$ is executed with probability $1/8$ and $b' = \mathsf{false}$ and S is executed with probability $1/8$;
  - $b''_0 = v$ with probability $1/4$, and in this case $b' = \mathsf{false}$ and S is executed.

So

$$\begin{aligned}\Pr[C'[Q \mid R_x] : \mathsf{S}/b_0 = v] &= \frac{5}{8}\Pr[C[Q^\circ_v] : D = v] + \frac{3}{8}\Pr[C[Q^\circ_v] : D \neq v]\\ &= \frac{3}{8} + \frac{1}{4}\Pr[C[Q^\circ_v] : D = v]\end{aligned}$$

because $\Pr[C[Q^\circ_v] : D \neq v] = 1 - \Pr[C[Q^\circ_v] : D = v]$. Finally, since $C'[Q \mid R_x]$ always executes either S or $\overline{\mathsf{S}}$, we obtain

$$\begin{aligned}&\Pr[C'[Q \mid R_x] : \mathsf{S}] - \Pr[C'[Q \mid R_x] : \overline{\mathsf{S}}]\\ &\quad = 2.\Pr[C'[Q \mid R_x] : \mathsf{S}] - 1\\ &\quad = \Pr[C'[Q \mid R_x] : \mathsf{S}/b_0 = 0] + \Pr[C'[Q \mid R_x] : \mathsf{S}/b_0 = 1] - 1\\ &\quad = \frac{3}{8} + \frac{1}{4}\Pr[C[Q^\circ_0] : D = 0] + \frac{3}{8} + \frac{1}{4}\Pr[C[Q^\circ_1] : D = 1] - 1\\ &\quad = \frac{1}{4}(1 - \Pr[C[Q^\circ_0] : D]) + \frac{1}{4}\Pr[C[Q^\circ_1] : D] - \frac{1}{4}\\ &\quad = \frac{1}{4}(\Pr[C[Q^\circ_1] : D] - \Pr[C[Q^\circ_0] : D])\end{aligned}$$

so $\Pr[C[Q^\circ_1] : D] - \Pr[C[Q^\circ_0] : D] = 4(\Pr[C'[Q \mid R_x] : \mathsf{S}] - \Pr[C'[Q \mid R_x] : \overline{\mathsf{S}}]) \leq 4p(C + t_D) = p'(C, t_D)$ $\square$

B Details on the Proof of TLS

This appendix relies on the CryptoVerif analysis of TLS by Bhargavan et al. described in [10, Section 6]. The CryptoVerif scripts for their analysis are available at `https://github.com/Inria-Prosecco/reftls/tree/master/cv`. We provide a brief reminder of the results of this analysis, before explaining the composition. For the full details, we still recommend reading [10, Section 6] before this appendix.

B.1 Reminder

Even though CryptoVerif can evaluate the probability of success of an attack as a function of the number of sessions and the probability of breaking each primitive (exact security), for simplicity, as in [10], we consider the asymptotic framework in which we only show that the probability of success of an attack is negligible as a function of the security parameter η. (A function f is *negligible* when for all polynomials q, there exists $\eta_o \in \mathbb{N}$ such that for all $\eta > \eta_0$, $f(\eta) \leq \frac{1}{q(\eta)}$.) Therefore, we omit probabilities from the security properties. All processes run in polynomial time in the security parameter and manipulate bitstrings of polynomially bounded length.

The analysis of [10, Section 6] splits TLS 1.3 into three components: the initial handshake, the handshakes with pre-shared key, and the record protocol.

Initial Handshake The initial handshake is a Diffie-Hellman key-exchange protocol between a client and a server. It provides 4 keys: the server application traffic secret *sats* used by the record protocol for sending messages from the server to the client, the client application traffic secret *cats* similar for messages from the client to the server, the exporter master secret *ems*, and the resumption secret *resumption_secret* used as pre-shared key by the next handshake. The key *sats* can be used to send messages from the server to the client before the last message of the handshake; these messages are called 0.5-RTT messages. These keys are stored in variables with prefix *c_* on the client side and *s_* on the server side. Furthermore, on the server side, the variables *s_cats*, *s_ems*, and *s_resumption_secret* are duplicated, with suffixes 1 and 2, for a technical reason (see [10]). CryptoVerif proves the following correspondences:

$$\textbf{inj-event}(\mathsf{ClientTerm}(log_4, s\_keys)) \Longrightarrow \textbf{inj-event}(\mathsf{ServerAccept}(log_4, s\_keys, i)) \tag{30}$$

$$\textbf{event}(\mathsf{ServerAccept}(log_4, s\_keys, i)) \wedge \textbf{event}(\mathsf{ServerAccept}(log_4, s\_keys', i')) \Longrightarrow i = i' \tag{31}$$

$$\textbf{inj-event}(\mathsf{ServerTerm}(log_7, c\_keys)) \Longrightarrow \textbf{inj-event}(\mathsf{ClientAccept}(log_7, c\_keys, i)) \tag{32}$$

$$\textbf{event}(\mathsf{ClientAccept}(log_7, c\_keys, i)) \wedge \textbf{event}(\mathsf{ClientAccept}(log_7, c\_keys', i')) \Longrightarrow i = i' \tag{33}$$

with public variables $V_{\text{in}} = \{c\_cats,\ c\_sats,\ c\_ems,\ c\_resumption\_secret,\ s\_cats1,\ s\_cats2,\ s\_sats,\ s\_ems1,\ s\_ems2,\ s\_resumption\_secret1,\ s\_resumption\_secret2\}$. The correspondences (30) and (32) prove injective mutual key authentication, and (31) and (33) prove that the accept events are executed at most once for value of the session identifier log_4 or log_7. The session identifier log_7 contains all messages of the protocol, while log_4 contains all messages except the last one (client `Finished`). The keys *c_keys* include all four keys mentioned above, while *s_keys* does not contain the resumption secret. CryptoVerif also proves that the protocol preserves the secrecy of *s_sats* with public variables $V_{\text{in}} \setminus \{c\_sats, s\_sats\}$, of *c_cats* with public variables $V_{\text{in}} \setminus \{c\_cats, s\_cats1, s\_cats2\}$, of *c_ems* with public variables $V_{\text{in}} \setminus \{c\_ems, s\_ems1, s\_ems2\}$, and of *c_resumption_secret* with public variables $V_{\text{in}} \setminus \{c\_resumption\_secret, s\_resumption\_secret1, s\_resumption\_secret2\}$. The events are executed and the keys are stored in the corresponding variable only when each participant believes that it talks to a honest peer: when the honest peer is authenticated and not compromised, or when the honest peer is compromised but the received messages still come from it, or when the peer is not authenticated but the Diffie-Hellman share comes from the honest peer.

Handshakes with Pre-Shared Key The handshakes with pre-shared key rely on a previously established pre-shared key to execute a handshake between the client and the server, with or without Diffie-Hellman key exchange. They produce the same keys as the initial handshake, plus an additional client early traffic secret *cets*, computed after the first message of the protocol (`ClientHello`). This traffic secret is used by the record protocol to send messages from the client to the server immediately after the `ClientHello` message, so-called 0-RTT data. These keys are stored in variables with prefix *c_* client side and *s_* server side for the handshake without Diffie-Hellman exchange, and *cdhe_* client side and *sdhe_* server side for the handshake with Diffie-Hellman exchange. Server-side, *cets* is stored in the variables *s_cets2* when the `ClientHello` has not been altered by an adversary, *s_cets3* when it has been altered, and *s_cets1* when it has been altered and the considered `ClientHello` message has not been received before (and similarly with *dhe* added for the handshake with Diffie-Hellman exchange). CryptoVerif proves the correspondences (30) to (33) with public variables V_{psk} containing all variables with prefixes *c_*, *s_*, *cdhe_*, and *sdhe_*. CryptoVerif also proves that the protocol preserves the secrecy of *s_sats* with public variables $V_{\text{psk}} \setminus \{c\_sats, s\_sats\}$, of *c_cats* with public variables $V_{\text{psk}} \setminus \{c\_cats, s\_cats\}$, of *c_ems* with public variables $V_{\text{psk}} \setminus \{c\_ems, s\_ems\}$, and of *c_resumption_secret* with public variables $V_{\text{psk}} \setminus \{c\_resumption\_secret, s\_resumption\_secret\}$.

For 0-RTT traffic, CryptoVerif shows the correspondences

$$\textbf{event}(\mathsf{ServerEarlyTerm1}(\mathit{log1}, \mathit{cets})) \Longrightarrow \textbf{event}(\mathsf{ClientEarlyAccept1}(\mathit{log1}, \mathit{cets}, i)) \tag{34}$$

$$\textbf{event}(\mathsf{ClientEarlyAccept1}(\mathit{log1}, \mathit{cets}, i)) \wedge \textbf{event}(\mathsf{ClientEarlyAccept1}(\mathit{log1}, \mathit{cets}', i')) \Longrightarrow i = i' \tag{35}$$

with public variables V_{psk}, and the secrecy of *c_cets* with public variables $V_{\text{psk}} \setminus \{c\_cets, s\_cets2\}$. These properties deal with the case of `ClientHello` messages that have not been altered by the adversary. The authentication (34) is non-injective because the `ClientHello` message can be replayed.

To deal with the case of altered `ClientHello` messages, CryptoVerif shows the correspondence

$$\textbf{event}(\mathsf{ServerEarlyTerm2}(\mathit{log1}, \mathit{cets})) \wedge \textbf{event}(\mathsf{ServerEarlyTerm2}(\mathit{log1}, \mathit{cets}')) \Longrightarrow \mathit{cets} = \mathit{cets}' \tag{36}$$

with public variables V_{psk}, and the secrecy of *s_cets1* with public variables $V_{\text{psk}} \setminus \{s\_cets1, s\_cets3\}$. The correspondence (36) means that, if the server receives twice the same altered `ClientHello` message, then it computes the same early traffic secret *cets*.

CryptoVerif proves similar properties for the handshake with Diffie-Hellman exchange, with suffix DHE added to the events and *dhe* added to the variables.

Record Protocol The record protocol is modeled in CryptoVerif as follows:

$$\begin{aligned}\mathit{Rec} = c_1(); \textbf{new}\ b : \mathit{bool}; \textbf{new}\ \mathit{ts} : \mathit{key}; \textbf{let}\ \mathit{ts}_{upd} = \mathsf{HKDF\_expand\_upd\_label}(\mathit{ts})\ \textbf{in}\ \overline{c_2}\langle\rangle;\\ (Q_{send}(b) \mid Q_{recv})\end{aligned}$$

It chooses a random bit b ($\mathsf{false} = 0$ or $\mathsf{true} = 1$) and a random traffic secret ts. It computes the updated traffic secret ts_{upd}, and then provides two processes $Q_{send}(b)$ and Q_{recv}. The process $Q_{send}(b)$ receives two clear messages msg_0 and msg_1, and a counter *count*. Provided the counter has not been used for sending a previous message and the messages msg_0 and msg_1 have the same padded length, it executes the event $\mathsf{sent}_0(count, msg_b)$ and sends the message msg_b encrypted using keys derived from the traffic secret ts. The process Q_{recv} receives an encrypted message and a counter $count$. Provided the counter has not been used for receiving a previous message, it decrypts the message using keys derived from the traffic secret ts and executes event $\mathsf{received}_0(count, msg)$ where msg is the clear message. Both the emission and reception can be executed several times, and the encryption scheme is authenticated.

CryptoVerif proves the secrecy of ts_{upd} with public variable b, the secrecy of b with public variable ts_{upd}, and the correspondence

$$\textbf{inj-event}(\mathsf{received}_0(count, msg)) \Longrightarrow \textbf{inj-event}(\mathsf{sent}_0(count, msg))$$

with public variables b, ts_{upd}. This correspondence shows injective message authentication.

We consider two other variants of the record protocol, used for 0-RTT. In the first variant, $Rec_{\text{0-RTT}}$, the receiver process is replicated once more, so that several sessions may have the same traffic secret, thus the receiver accepts messages with the same counter in different sessions with the same traffic secret. It models that the server may receive several times the same `ClientHello` message, yielding the same traffic secret. In this model, CryptoVerif proves the secrecy of ts_{upd} with public variable b, the secrecy of b with public variable ts_{upd}, and the correspondence

$$\textbf{event}(\mathsf{received}_0(count, msg)) \Longrightarrow \textbf{event}(\mathsf{sent}_0(count, msg))$$

with public variables b, ts_{upd}.

In the second variant, $Rec_{\text{0-RTT,Bad}}$, the sender process is additionally removed. This model corresponds to the situation in which the `ClientHello` message is altered, and thus the server obtains a traffic secret that is not used by any client. In this model, CryptoVerif proves the secrecy of ts_{upd} with no public variable and the correspondence $\textbf{event}(\mathsf{received}_0(count, msg)) \Longrightarrow \mathsf{false}$ with public variable ts_{upd}, that is, $\textbf{event}(\mathsf{received}_0(count, msg))$ can be executed only with negligible probability.

B.2 The Composition

Let us now apply our composition theorems to compose the three parts of TLS 1.3 into the whole protocol.

Record Protocol with Key Updates We compose the record protocol with itself, to deal with key updates.

As a first step, let us establish properties of the record protocol without composition. For $s \in \{0, 1\}$, let Rec_s be the protocol obtained by always choosing $b = s$ in the record protocol Rec. Let Rec_s° be obtained by removing events in Rec_s. Let $Rec_\star$ be the protocol in which the choice of b is removed and the sender receives a single clear message and sends it encrypted. We let $corr_j$ be the correspondence

$$\textbf{inj-event}(\mathsf{received}_j(count, msg)) \Longrightarrow \textbf{inj-event}(\mathsf{sent}_j(count, msg))$$

We prove that $Rec_0^\circ \approx^V Rec_1^\circ$ and for all $s \in \{0, 1, \star\}$, Rec_s preserves the secrecy of ts_{upd} and satisfies the correspondence $corr_0$ with public variables V, where $V = \{ts_{upd}\}$.

Proof. By Lemma 12, since Rec preserves the secrecy of b with public variables V, we have $Rec_0^\circ \approx^V Rec_1^\circ$.

Moreover, since Rec preserves the secrecy of ts_{upd} and satisfies the correspondence $corr_0$ with public variables V, so do Rec_0 and Rec_1 by Lemma 1, because for $s \in \{0, 1\}$, $Rec_s \approx^V C_s[Rec]$

and the evaluation context C_s receives two messages msg_0 and msg_1 and forwards two copies of msg_s to Rec, so that Rec always sends msg_s encrypted independently of the value of b.

Furthermore, since Rec preserves the secrecy of ts_{upd} and satisfies the correspondence $corr_0$ with public variables V, $Rec_\star$ satisfies the same properties by Lemma 1, because $Rec_\star \approx^V C[Rec]$, where the evaluation context C receives a single message msg and forwards two copies of msg to Rec, so that Rec always sends $msg_0 = msg_1 = msg$ encrypted independently of the value of b. □

Let us now define the record protocol with m key updates as follows:

$$Rec_s^m = c_1(); \mathbf{new}\ ts : key; \overline{c_2}\langle\rangle; (\prod_{j=0}^m Q^j_{send,s} \mid \prod_{j=0}^m Q^j_{recv})$$

where s is the side (0, 1, or $\star$), $Q^j_{send,s}$ is a process that receives messages on channel c_3^j and sends them encrypted using the j-th updated traffic secret on channel c_4^j, and Q^j_{recv} is a process that receives messages on channel c_5^j and decrypts them using the j-th updated traffic secret. The process $Q^j_{send,s}$ uses table $table\_count\_send^j$ and event sent_j; Q^j_{recv} uses table $table\_count\_recv^j$ and event $\mathsf{received}_j$. When s is 0 or 1, $Q^j_{send,s}$ receives two clear messages msg_0 and msg_1 and sends msg_s encrypted. When $s = \star$, $Q^j_{send,s}$ receives a single message and sends it encrypted. Let $Rec_s^{m,\circ}$ be obtained by removing all events in Rec_s^m.

The record protocol can be written:

$$\begin{aligned} Rec_s = c_1(); \mathbf{new}\ ts : key; \mathbf{let}\ ts_{upd} = \mathsf{HKDF\_expand\_upd\_label}(ts)\ \mathbf{in}\ \overline{c_2}\langle\rangle;\\ (Q^0_{send,s} \mid Q^0_{recv}) \end{aligned}$$

We prove that $Rec_0^{m,\circ} \approx Rec_1^{m,\circ}$ and for all $s \in \{0, 1, \star\}$, Rec_s^m satisfies the correspondences $corr_j$ with no public variable for $j \leq m$.

Proof. The proof proceeds by induction on m.

Case $m = 0$: Since $Rec_0^\circ \approx^V Rec_1^\circ$ and for all $s \in \{0, 1, \star\}$, Rec_s satisfies the correspondence $corr_0$ with public variables ts_{upd}, we have $Rec_0^{0,\circ} \approx Rec_1^{0,\circ}$ and for all $s \in \{0, 1, \star\}$, Rec_s^0 satisfies the correspondence $corr_0$ with no public variable, by removing the assignment to ts_{upd}.

Case $m+1$: By induction hypothesis, $Rec_0^{m,\circ} \approx Rec_1^{m,\circ}$ and for all $s \in \{0, 1, \star\}$, Rec_s^m satisfies the correspondences $corr_j$ with no public variable for $j \leq m$. Let ${Rec'}_s^m$ be obtained from Rec_s^m by renaming ts to ts_{upd} and other variables to fresh variables, the events $\mathsf{received}_j$ and sent_j to $\mathsf{received}_{j+1}$ and sent_{j+1} respectively for $j \leq m$, the channels c_1, c_2, c_3^j, c_4^j, c_5^j to c'_1, c'_2, c_3^{j+1}, c_4^{j+1}, c_5^{j+1} respectively for $j \leq m$, and the tables $table\_count\_send^j$ and $table\_count\_recv^j$ to $table\_count\_send^{j+1}$ and $table\_count\_recv^{j+1}$ respectively for $j \leq m$. The processes ${Q'}^j_{send,s}$ and ${Q'}^j_{recv}$ are defined in a similar way.

We compose Rec_s with ${Rec'}_{s'}^m$ by Theorem 1. By the previous renaming, ts_{upd} is the only common variable between the processes Rec_s and ${Rec'}_{s'}^m$, and these processes have no common channel, no common event, and no common table. We obtain the composed system

$$\begin{aligned} S_{composed,s,s'} = c_1(); \mathbf{new}\ ts : key; \mathbf{let}\ ts_{upd} = \mathsf{HKDF\_expand\_upd\_label}(ts)\ \mathbf{in}\ \overline{c''_2}\langle\rangle;\\ (Q^0_{send,s} \mid Q^0_{recv} \mid \prod_{j=0}^m {Q'}^j_{send,s'} \mid \prod_{j=0}^m {Q'}^j_{recv}) \end{aligned}$$

for some fresh channel c''_2, so that we have

$$S_{composed,s,s}\{c_2/c''_2\} \approx Rec_s^{m+1}$$

Let $S^{\circ}_{composed,s,s'}$ be obtained from $S_{composed,s,s'}$ by removing events $\mathsf{received}_0$ and sent_0, and $S^{\circ\circ}_{composed,s,s'}$ be obtained from $S_{composed,s,s'}$ by removing all events. Since Rec_s preserves the secrecy of ts_{upd}, Theorem 1 shows that $S^{\circ}_{composed,s,s'} \approx C'_s[{Rec'_{s'}}^m]$ for some evaluation context C'_s acceptable for ${Rec'_{s'}}^m$ without public variables and containing no event. Moreover, it also shows that $S_{composed,s,s'} \approx C''_{s'}[Rec_s]$ for some evaluation context $C''_{s'}$ acceptable for Rec_s with public variable ts_{upd} and containing the events $\mathsf{received}_j$ and sent_j for $1 \leq j \leq m+1$. Indeed, C'_s does not depend on the side s' of ${Rec'_{s'}}^m$ and $C''_{s'}$ does not depend on the side s of Rec_s.

Since $S_{composed,s,s} \approx C''_s[Rec_s]$ and Rec_s satisfies the correspondence $corr_0$ with public variables ts_{upd}, then by Lemma 1, $S_{composed,s,s}$ satisfies the correspondence $corr_0$ with no public variable, and so does Rec_s^{m+1}. Since

$$S^{\circ}_{composed,s,s} \approx C'_s[{Rec'_s}^m]$$

nd ${Rec'_s}^m$ satisfies the correspondences $corr_j$ with no public variable for $1 \leq j \leq m+1$, then by Lemma 1 so do $S^{\circ}_{composed,s,s}$, $S_{composed,s,s}$, and Rec_s^{m+1}. Hence Rec_s^{m+1} satisfies the correspondences $corr_j$ with no public variable for $j \leq m+1$.

Finally, let C''°_0 be obtained from C''_0 be removing the events $\mathsf{received}_j$ and sent_j for $1 \leq j \leq m+1$. Using Lemma 11,

$$\begin{aligned} S^{\circ\circ}_{composed,0,0} &\approx C''^{\circ}_0[Rec^{\circ}_0] \approx C''^{\circ}_0[Rec^{\circ}_1] \approx S^{\circ\circ}_{composed,1,0} \\ &\approx C'_1[{Rec'_0}^{m,\circ}] \approx C'_1[{Rec'_1}^{m,\circ}] \approx S^{\circ\circ}_{composed,1,1} \end{aligned} \tag{37}$$

so $Rec_0^{m+1,\circ} \approx Rec_1^{m+1,\circ}$ by renaming c''_2 to c_2. □

We can apply a similar reasoning to the two variants of the record protocol for 0-RTT data.

- For $s \in \{0,1\}$, let ${Rec_{\text{0-RTT}}}_s^m$ be the process obtained by always choosing $b = s$ in $Rec_{\text{0-RTT}}$ and composing it with itself m times. Let ${Rec_{\text{0-RTT}}}_\star^m$ be the process $Rec_{\text{0-RTT}}$ in which the choice of b is removed and the sender receives a single clear message and sends it encrypted, composed with itself m times. Let ${Rec_{\text{0-RTT}}}_s^{m,\circ}$ be obtained from ${Rec_{\text{0-RTT}}}_s^m$ by removing all events. We prove that ${Rec_{\text{0-RTT}}}_0^{m,\circ} \approx {Rec_{\text{0-RTT}}}_1^{m,\circ}$ and that, for all $s \in \{0,1,\star\}$, ${Rec_{\text{0-RTT}}}_s^m$ satisfies the correspondences

$$\mathbf{event}(\mathsf{received}_j(count, msg)) \Longrightarrow \mathbf{event}(\mathsf{sent}_j(count, msg))$$

  with no public variable for all $j \leq m$.

- We also prove that the process ${Rec_{\text{0-RTT,Bad}}}^m$, obtained by composing $Rec_{\text{0-RTT,Bad}}$ with itself m times, satisfies the correspondences

$$\mathbf{event}(\mathsf{received}_j(count, msg)) \Longrightarrow \mathsf{false}$$

  with no public variable for all $j \leq m$. (This variant contains no sender process, so it does not use b.)

By Lemmas 5 and 6, we infer that replicated versions of the record protocol satisfy similar properties. Let us define $Repl_C(Q) = \mathsf{AddReplSid}(i_C \leq n_C, c'_1, bitstring, Q)$ and $Repl_S(Q) = \mathsf{AddReplSid}(i_S \leq n_S, c'_1, bitstring, Q)$.

- $Repl_C(Rec_0^{m,\circ}) \approx Repl_C(Rec_1^{m,\circ})$ and for all $s \in \{0,1,\star\}$, $Repl_C(Rec_s^m)$ satisfies the correspondences $\mathbf{inj\text{-}event}(\mathsf{received}_j(x, count, msg)) \Longrightarrow \mathbf{inj\text{-}event}(\mathsf{sent}_j(x, count, msg))$ with no public variable for all $j \leq m$.

- $\mathit{Repl}_S(\mathit{Rec}_s^m)$ satisfies the same properties.

- $\mathit{Repl}_C(\mathit{Rec}_{\text{0-RTT}}{}_0^{m,\circ}) \approx \mathit{Repl}_C(\mathit{Rec}_{\text{0-RTT}}{}_1^{m,\circ})$ and for all $s \in \{0, 1, \star\}$, $\mathit{Repl}_C(\mathit{Rec}_{\text{0-RTT}}{}_s^m)$ satisfies the correspondences $\mathbf{event}(\mathsf{received}_j(x, \mathit{count}, \mathit{msg})) \implies \mathbf{event}(\mathsf{sent}_j(x, \mathit{count}, \mathit{msg}))$ with no public variable for all $j \leq m$.

- $\mathit{Repl}_S(\mathit{Rec}_{\text{0-RTT,Bad}}{}^m)$ satisfies the correspondences $\mathbf{event}(\mathsf{received}_j(x, \mathit{count}, \mathit{msg})) \implies \mathsf{false}$ with no public variable for all $j \leq m$.

Handshakes with Pre-Shared Key We compose the handshakes with pre-shared key with the record protocol:

1. with secret key *c_cats* using Theorem 2 and process $\mathit{Repl}_C(\mathit{Rec}_s^m)$, for 1-RTT client-to-server messages. (Event e_A is ClientAccept and event e_B is ServerTerm. In [10, Figure 9], event ClientAccept occurs at line 3: and event ServerTerm occurs at line 7:.)

2. with secret key *s_sats* using Theorem 2 and process $\mathit{Repl}_S(\mathit{Rec}_s^m)$, for 0.5-RTT and 1-RTT server-to-client messages. (Event e_A is ServerAccept and event e_B is ClientTerm. In [10, Figure 9], event ServerAccept occurs at line 6: and event ClientTerm occurs at line 2:.)

3. with secret key *c_cets* using Theorem 3 and the variant $\mathit{Repl}_C(\mathit{Rec}_{\text{0-RTT}}{}_s^m)$ of the record protocol with an additional replication in the receiver process, for 0-RTT messages when the `ClientHello` message has not been altered. (Event e_A is ClientEarlyAccept1 and event e_B is ServerEarlyTerm1. In [10, Figure 9], event ClientEarlyAccept1 occurs at line 1: and event ServerEarlyTerm1 occurs at line 4:.)

4. with secret key *s_cets3* using Theorem 5 and the variant $\mathit{Repl}_S(\mathit{Rec}_{\text{0-RTT,Bad}}{}^m)$ of the record protocol without sender process, for 0-RTT when the `ClientHello` message has been altered. (Event e is ServerEarlyTerm2, which occurs at line 5: in [10, Figure 9].)

We perform similar compositions (numbered 1' to 4') in the handshake with pre-shared key and Diffie-Hellman key agreement.

Before applying the theorems, we permute assignments and events as needed so that the process has the form required in the theorem. We also notice that the events contain more keys in the process than in the composition theorems: the handshake process contains events $e_A((\widetilde{msg}_A), (\widetilde{k}_A), \widetilde{i})$ and $e_B((\widetilde{msg}_B), (\widetilde{k}_B))$ while in the composition theorems, the process Q'_1 contains $e_A((\widetilde{msg}_A), k_A, \widetilde{i})$ and $e_B((\widetilde{msg}_B), k_B)$, where $\widetilde{k}_A$ contains k_A and $\widetilde{k}_B$ contains k_B at the same position. (The function sid just returns the tuple of messages.) As a result, the correspondences (2) and (7) that we prove are stronger than those in the composition theorems: they mean that for each execution of event $e_B((\widetilde{msg}), (\widetilde{k}))$, there is an execution of event $e_A((\widetilde{msg}), (\widetilde{k}), \widetilde{i})$ for some $\widetilde{i}$, with the same tuples of keys $\widetilde{k} = \widetilde{k}_A = \widetilde{k}_B$ while the composition theorems just require the equality for one key $k = k_A = k_B$. The meaning of the correspondence (3) is unchanged, since in this correspondence, the keys are distinct variables so their value does not matter.

In the composition, we index the events that come from the record protocol with an index c from 1 to 4 and from 1' to 4' corresponding to the number of the composition that adds this instance of the record protocol, so we use events $\mathsf{sent}_{c,j}$ and $\mathsf{received}_{c,j}$. We index the channels, tables, and variables from the record protocol in a similar way, so that they are distinct in the various compositions.

For the first composition, we compose the handshakes with pre-shared key Q_{PSKH} with the record protocol $\mathit{Rec}_s^{1,m}$, obtained from $\mathit{Repl}_C(\mathit{Rec}_s^m)$ by this renaming, with secret key *c_cats* using Theorem 2, as announced above. We obtain a composed protocol $S^{1,m}_{composed,s}$. Let $S^{1,m,\circ}_{composed,s}$ be obtained from $S^{1,m}_{composed,s}$ by removing the events of the handshake protocol, and $S^{1,m,\circ\circ}_{composed,s}$ be obtained from $S^{1,m}_{composed,s}$ by removing all events. By Theorem 2, we have $S^{1,m,\circ}_{composed,s} \overset{\sim V^1_{\mathrm{psk}},\emptyset}{\rightarrow}_{f^1} Rec^{1,m}_s$ and $S^{1,m}_{composed,s} \approx^{V^1_{\mathrm{psk}}} C^1_s[Q_{\mathrm{PSKH}}]$ for some evaluation context C^1_s acceptable for Q_{PSKH} with public variables $\{c\_cats, s\_cats\}$, where $V^1_{\mathrm{psk}} = V_{\mathrm{psk}} \setminus \{c\_cats, s\_cats\}$. By Lemma 11, we have $S^{1,m,\circ\circ}_{composed,s} \overset{\sim V^1_{\mathrm{psk}},\emptyset}{\rightarrow}_{f^1} Rec^{1,m,\circ}_s$, where the process $Rec^{1,m,\circ}_s$ is obtained by deleting events in $Rec^{1,m}_s$. Since $Repl_C(Rec^{m,\circ}_0) \approx Repl_C(Rec^{m,\circ}_1)$, that is, $Rec^{1,m,\circ}_0 \approx Rec^{1,m,\circ}_1$, we have $S^{1,m,\circ\circ}_{composed,0} \approx^{V^1_{\mathrm{psk}}} S^{1,m,\circ\circ}_{composed,1}$ by Lemma 2. By Lemma 3 and $S^{1,m,\circ}_{composed,s} \overset{\sim V^1_{\mathrm{psk}},\emptyset}{\rightarrow}_{f^1} Rec^{1,m}_s$, the processes $S^{1,m,\circ}_{composed,s}$ and $S^{1,m}_{composed,s}$ inherit the correspondence properties of the record protocol. By Lemma 1 and $S^{1,m}_{composed,s} \approx^{V^1_{\mathrm{psk}}} C_s[Q_{\mathrm{PSKH}}]$, the process $S^{1,m}_{composed,s}$ inherits correspondence and secrecy properties of the handshake with pre-shared keys (but the variable *c_cats* on which we compose and the corresponding variable *s_cats* in the peer are removed from the public variables V^1_{psk} and from the secrets).

We perform the second composition in a similar way. We compose $S^{1,m}_{composed,s}$ with the record protocol $Rec^{2,m}_{s'}$, obtained by renaming $Repl_S(Rec^m_{s'})$, with secret key *s_sats* using Theorem 2, as announced above. We obtain a composed protocol $S^{2,m}_{composed,s,s'}$. Let $S^{2,m,\circ}_{composed,s,s'}$ be obtained from $S^{2,m}_{composed,s,s'}$ by removing the events of $S^{1,m}_{composed,s}$, and $S^{2,m,\circ\circ}_{composed,s,s'}$ be obtained from $S^{2,m}_{composed,s,s'}$ by removing all events. By Theorem 2, we have $S^{2,m,\circ}_{composed,s,s'} \overset{\sim V^2_{\mathrm{psk}},\emptyset}{\rightarrow}_{f^2_s} Rec^{2,m}_{s'}$ and $S^{2,m}_{composed,s,s'} \approx^{V^2_{\mathrm{psk}}} C^2_{s'}[S^{1,m}_{composed,s}]$ for some evaluation context $C^2_{s'}$ acceptable for $S^{1,m}_{composed,s}$ with public variables $\{s\_sats, c\_sats\}$, where $V^2_{\mathrm{psk}} = V^1_{\mathrm{psk}} \setminus \{s\_sats, c\_sats\}$. By Lemma 11, we have $S^{2,m,\circ\circ}_{composed,s,s'} \overset{\sim V^2_{\mathrm{psk}},\emptyset}{\rightarrow}_{f^2_s} Rec^{2,m,\circ}_{s'}$, where the process $Rec^{2,m,\circ}_{s'}$ is obtained by deleting events in $Rec^{2,m}_{s'}$. Since $Repl_S(Rec^{m,\circ}_0) \approx Repl_S(Rec^{m,\circ}_1)$, that is, $Rec^{2,m,\circ}_0 \approx Rec^{2,m,\circ}_1$, we have

$$S^{2,m,\circ\circ}_{composed,0,0} \approx^{V^2_{\mathrm{psk}}} S^{2,m,\circ\circ}_{composed,0,1}$$

by Lemma 2. Moreover, using Lemma 11,

$$S^{2,m,\circ\circ}_{composed,0,1} \approx^{V^2_{\mathrm{psk}}} C^{2,\circ}_1[S^{1,m,\circ\circ}_{composed,0}] \approx^{V^2_{\mathrm{psk}}} C^{2,\circ}_1[S^{1,m,\circ\circ}_{composed,1}] \approx^{V^2_{\mathrm{psk}}} S^{2,m,\circ\circ}_{composed,1,1}$$

so

$$S^{2,m,\circ\circ}_{composed,0,0} \approx^{V^2_{\mathrm{psk}}} S^{2,m,\circ\circ}_{composed,1,1}\,.$$

By Lemma 3 and $S^{2,m,\circ}_{composed,s,s} \overset{\sim V^2_{\mathrm{psk}},\emptyset}{\rightarrow}_{f^2_s} Rec^{2,m}_s$, the processes $S^{1,m,\circ}_{composed,s,s}$ and $S^{1,m}_{composed,s,s}$ inherit the correspondence properties of the record protocol. By Lemma 1 and $S^{2,m}_{composed,s,s} \approx^{V^2_{\mathrm{psk}}} C^2_s[S^{1,m}_{composed,s}]$, the process $S^{2,m}_{composed,s,s}$ inherits correspondence and secrecy properties of the handshake with pre-shared keys (but the variable *s_sats* on which we compose and the corresponding variable *c_sats* in the peer are removed from the public variables V^2_{psk} and from the secrets).

We perform the other compositions in the same way. We obtain processes $Q^m_{\mathrm{PSKH},s}$ that run one handshake with pre-shared key and the record protocol with at most m key updates and side s. Let $Q^{m,\circ}_{\mathrm{PSKH},s}$ be obtained from $Q^m_{\mathrm{PSKH},s}$ by removing all events. The composition theorems show that $Q^{m,\circ}_{\mathrm{PSKH},0} \approx^V Q^{m,\circ}_{\mathrm{PSKH},1}$ where $V = \{c\_ems, s\_ems,$ $cdhe\_ems, sdhe\_ems, c\_resumption\_secret, s\_resumption\_secret, cdhe\_resumption\_secret,$ $sdhe\_resumption\_secret\}$, and that $Q^m_{\mathrm{PSKH},s}$ satisfies correspondences and secrecy properties that come from the handshake (but the variables c_cats, s_sats, c_cets, s_cets3, $cdhe\_cats$, $sdhe\_sats$, $cdhe\_cets$, and $sdhe\_cets3$ on which we compose and the corresponding variables in the peer are removed from the public variables V and from the secrets) and from the record protocol (with additional index c in the correspondences). That is, $Q^m_{\mathrm{PSKH},s}$ satisfies the correspondences (30) to (36), the correspondences (30) to (36) with suffix DHE in the events, and

$$\begin{aligned}
&\textbf{inj-event}(\mathsf{received}_{c,j}(x, count, msg)) \Longrightarrow \textbf{inj-event}(\mathsf{sent}_{c,j}(x, count, msg))\\
&\qquad \text{for } c \in \{1, 2, 1', 2'\} \text{ and } j \le m\\
&\textbf{event}(\mathsf{received}_{c,j}(x, count, msg)) \Longrightarrow \textbf{event}(\mathsf{sent}_{c,j}(x, count, msg))\\
&\qquad \text{for } c \in \{3, 3'\} \text{ and } j \le m\\
&\textbf{event}(\mathsf{received}_{c,j}(x, count, msg)) \Longrightarrow \mathsf{false}\\
&\qquad \text{for } c \in \{4, 4'\} \text{ and } j \le m
\end{aligned}$$

with public variables V and preserves the secrecy of

$$\begin{aligned}
&c_ems \text{ with public variables } V \setminus \{c_ems, s_ems\},\\
&cdhe_ems \text{ with public variables } V \setminus \{cdhe_ems, sdhe_ems\},\\
&c_resumption_secret \text{ with public variables}\\
&\qquad V \setminus \{c_resumption_secret, s_resumption_secret\},\\
&cdhe_resumption_secret \text{ with public variables}\\
&\qquad V \setminus \{cdhe_resumption_secret, sdhe_resumption_secret\}
\end{aligned}$$

Let us define processes $Q^{l,m}_{\mathrm{PSKH},s}$ that run at most $l+1$ successive handshakes with pre-shared key and the record protocol with at most m key updates and side s. In these processes, we add an index k to all events: k is a bitstring of length at most l, where the length of k is the number of handshakes with pre-shared key made before the current one and the i-th bit of k is 1 if and only if the i-th handshake used a Diffie-Hellman exchange. The index k is also added to the variables c_ems, s_ems, $cdhe\_ems$, $sdhe\_ems$. We also add arguments $\widetilde{x}_k$ to events, where $\widetilde{x}_k$ is a sequence of variables that will contain the messages sent and received by the handshakes above the current one. The sequence $\widetilde{x}_k$ contains as many variables as the length of k. The processes $Q^{l,m}_{\mathrm{PSKH},s}$ start with a context C_{h} that defines a random oracle: $Q^{l,m}_{\mathrm{PSKH},s} = C_{\mathsf{h}}[Q^{l,m}_{\text{PSKH-no-ROM},s}]$. Hence, the corresponding replicated process constructed by Lemmas 5 and 6 is $Q^{l,m}_{\text{PSKH-Repl},s} = C'_{\mathsf{h}}[\mathsf{AddReplSid}(i_C \le n_C, c', bitstring, Q^{l,m}_{\text{PSKH-no-ROM},s})]$. Let $Q^{l,m,\circ}_{\mathrm{PSKH},s}$ be obtained from $Q^{l,m}_{\mathrm{PSKH},s}$ by removing all events, and $Q^{l,m,\circ}_{\text{PSKH-Repl},s}$ be obtained from $Q^{l,m}_{\text{PSKH-Repl},s}$ by removing all events.

We have $Q^{0,m}_{\mathrm{PSKH},s} = Q^m_{\mathrm{PSKH},s}$. Using a proof by induction as for the record protocol above, from properties of $Q^{l,m}_{\mathrm{PSKH},s}$, we infer properties of $Q^{l,m}_{\text{PSKH-Repl},s}$ by Lemmas 5 and 6 and we compose $Q^m_{\mathrm{PSKH},s}$ with $Q^{l,m}_{\text{PSKH-Repl},s}$, with secret $c\_resumption\_secret$ using Theorem 2. (Event e_A is $\mathsf{ClientAccept}$ and event e_B is $\mathsf{ServerTerm}$.) We perform a similar composition with secret $cdhe\_resumption\_secret$. Then, we obtain properties of $Q^{l+1,m}_{\mathrm{PSKH},s}$.

We obtain that $Q^{l,m,\circ}_{\mathrm{PSKH},0} \approx^{V_l} Q^{l,m,\circ}_{\mathrm{PSKH},1}$ and that $Q^{l,m}_{\mathrm{PSKH},s}$ satisfies the same correspondences as $Q^m_{\mathrm{PSKH},s}$, with additional index k of length at most l and additional arguments $\widetilde{x}_k$ in the events and with public variables V_l, and preserves the secrecy of c_ems_k with public variables $V_l \setminus \{c\_ems_k, s\_ems_k\}$ and of $cdhe\_ems_k$ with public variables $V_l \setminus \{cdhe\_ems_k, sdhe\_ems_k\}$, where $V_l = \bigcup_{k \text{ of length at most } l} \{c\_ems_k,\ s\_ems_k,\ cdhe\_ems_k,\ sdhe\_ems_k\}$.

We also obtain that $Q^{l,m,\circ}_{\text{PSKH-Repl},0} \approx^{V_l} Q^{l,m,\circ}_{\text{PSKH-Repl},1}$ and that $Q^{l,m}_{\text{PSKH-Repl},s}$ satisfies the same correspondences as $Q^{m}_{\text{PSKH},s}$, with additional index k of length at most l and additional arguments $x, \widetilde{x}_k$ in the events and with public variables V_l, and preserves the secrecy of c_ems_k with public variables $V_l \setminus \{c\_ems_k, s\_ems_k\}$ and of $cdhe\_ems_k$ with public variables $V_l \setminus \{cdhe\_ems_k, sdhe\_ems_k\}$.

Full Protocol We compose the initial handshake with the record protocol:

1. with secret key c_cats using Theorem 4 and process $\mathsf{AddReplSid}(i_C \leq n_C, c', \mathit{bitstring}, \mathit{Rec}^m_s)$, for 1-RTT client-to-server messages. (Event e_A is $\mathsf{ClientAccept}$ and event e_B is $\mathsf{ServerTerm}$. In [10, Figure 8], event $\mathsf{ClientAccept}$ occurs at line 2: and event $\mathsf{ServerTerm}$ occurs at line 6: and in the process $Q_{\mathit{ServerAfter0.5RTT2}}$. We need the variant with several holes of Theorem 2, shown in Appendix A.10, because event $\mathsf{ServerTerm}$ occurs twice.)

2. with secret key c_sats using Theorem 2 and process $\mathsf{AddReplSid}(i_S \leq n_S, c', \mathit{bitstring}, \mathit{Rec}^m_s)$, for 0.5-RTT and 1-RTT server-to-client messages. (Event e_A is $\mathsf{ServerAccept}$ and event e_B is $\mathsf{ClientTerm}$. In [10, Figure 8], event $\mathsf{ServerAccept}$ occurs at line 4: and event $\mathsf{ClientTerm}$ occurs at line 1:.)

In the composition, we index the events that come from the record protocol with an index c (1 or 2) corresponding to the number of the composition that adds this instance of the record protocol, so we use events $\mathsf{sent}_{c,j}$ and $\mathsf{received}_{c,j}$.

We also compose the initial handshake with the process $Q^{l,m}_{\text{PSKH-Repl},s}$ that runs handshakes with pre-shared key, with secret key $c\_resumption\_secret$ using Theorem 4. (Event e_A is $\mathsf{ClientAccept}$ and event e_B is $\mathsf{ServerTerm}$. Again, we need the variant with several holes of Theorem 2, because event $\mathsf{ServerTerm}$ occurs twice.)

Finally, we compose the obtained process with a process that runs the rest of the TLS protocol (record protocol and/or handshakes with pre-shared key) without any event or security claim, when the honest client talks to a dishonest server or conversely. In these cases, the model leaks the session keys, so we just put the process in a context that receives these keys and runs the rest of TLS, and perform the composition by eliminating communications (Appendix A.2). In [10, Figure 8], this situation occurs at lines 3:, 5:, 7:, and at a line similar to 7: in the process $Q_{\mathit{ServerAfter0.5RTT2}}$.

We obtain processes $Q^{l,m}_{\text{TLS},s}$ that run the initial handshake followed by at most l successive handshakes with pre-shared key and the record protocol with at most m key updates and side s. Let $Q^{l,m,\circ}_{\text{TLS},s}$ be obtained from $Q^{l,m}_{\text{TLS},s}$ by removing all events.

These processes satisfy the following properties: $Q^{l,m,\circ}_{\text{TLS},0} \approx^{V'_l} Q^{l,m,\circ}_{\text{TLS},1}$ and $Q^{l,m}_{\text{TLS},s}$ satisfies the correspondences (30) to (33) (inherited from the initial handshake), the correspondences (30) to (36) with additional index k and additional arguments $x, \widetilde{x}_k$ in the events, the correspondences (30) to (36) with suffix DHE, additional index k, and additional arguments $x, \widetilde{x}_k$ in the events,

$$\begin{aligned}&\textbf{inj-event}(\mathsf{received}_{c,j}(x', \mathit{count}, \mathit{msg})) \Longrightarrow \textbf{inj-event}(\mathsf{sent}_{c,j}(x', \mathit{count}, \mathit{msg}))\\&\qquad\text{for } c \in \{1, 2, 1', 2'\} \text{ and } j \leq m\end{aligned} \tag{38}$$

$$\begin{aligned}&\textbf{inj-event}(\mathsf{received}_{k,c,j}(x', \widetilde{x}_k, x, \mathit{count}, \mathit{msg})) \Longrightarrow \textbf{inj-event}(\mathsf{sent}_{k,c,j}(x', \widetilde{x}_k, x, \mathit{count}, \mathit{msg}))\\&\qquad\text{for } c \in \{1, 2, 1', 2'\} \text{ and } j \leq m\end{aligned} \tag{39}$$

$$\begin{aligned}&\textbf{event}(\mathsf{received}_{k,c,j}(x', \widetilde{x}_k, x, \mathit{count}, \mathit{msg})) \Longrightarrow \textbf{event}(\mathsf{sent}_{k,c,j}(x', \widetilde{x}_k, x, \mathit{count}, \mathit{msg}))\\&\qquad\text{for } c \in \{3, 3'\} \text{ and } j \leq m\end{aligned} \tag{40}$$

$$\begin{aligned}&\mathbf{event}(\mathsf{received}_{k,c,j}(x', \widetilde{x}_k, x, \mathit{count}, \mathit{msg})) \Longrightarrow \mathsf{false}\\ &\qquad \text{for } c \in \{4, 4'\} \text{ and } j \leq m\end{aligned} \tag{41}$$

all with public variables V_l' and preserves the secrecy of

$$\begin{aligned}&c\_ems \text{ with public variables } V_l' \setminus \{c\_ems, s\_ems1, s\_ems2\},\\ &c\_ems_k \text{ with public variables } V_l' \setminus \{c\_ems_k, s\_ems_k\},\\ &cdhe\_ems_k \text{ with public variables } V_l' \setminus \{cdhe\_ems_k, sdhe\_ems_k\}\end{aligned}$$

where

$$V_l' = \{c\_ems, s\_ems1, s\_ems2\} \cup \bigcup_{k \text{ of length at most } l} \{c\_ems_k, s\_ems_k, cdhe\_ems_k, sdhe\_ems_k\}\,.$$

(The events and variables with additional index k are considered different from the events and variables without that index, even when k is empty. Those without index k come from the initial handshake. Those with index k come from the handshake with pre-shared key.)

The equivalence $Q^{l,m,\circ}_{\mathrm{TLS},0} \approx^{V_l'} Q^{l,m,\circ}_{\mathrm{TLS},1}$ proves message secrecy: an adversary cannot distinguish whether the protocol encrypted the first or the second set of plaintexts. The correspondences (38) and (39) prove injective message authentication for 0.5-RTT and 1-RTT data, while the correspondences (40) prove non-injective message authentication for 0-RTT data: if a honest receiver is in a honest session (that is, it talks to a honest sender) and receives a message, then this message was sent by the honest sender talking to the honest receiver. Furthermore, the message has the same associated counter *count* on both sides. Other correspondences show (injective or non-injective) key authentication, and unique accept for the composed protocol. We also obtain the secrecy of the exporter master secrets computed by the various handshakes: the adversary cannot distinguish them from independent random values.

Remark The correspondence (31) with additional index k and additional arguments $x, \widetilde{x}_k$ in the events is

$$\begin{aligned}&\mathbf{event}(\mathsf{ServerAccept}_k(x, \widetilde{x}_k, log_4, s\_keys, i)) \wedge\\ &\quad \mathbf{event}(\mathsf{ServerAccept}_k(x, \widetilde{x}_k, log_4, s\_keys', i')) \Longrightarrow i = i'\end{aligned} \tag{42}$$

It shows that, when two executions of $\mathsf{ServerAccept}_k$ with the same log $(x, \widetilde{x}_k, log_4)$ from the beginning of the protocol always have the same final replication index i. However, the intuitive extension of (31) to the composed system is rather that $\mathsf{ServerAccept}_k$ is executed at most once with the same log, that is, two executions of $\mathsf{ServerAccept}_k$ with the same log have *all* their replication indices equal:

$$\begin{aligned}&\mathbf{event}(\mathsf{ServerAccept}_k(x, \widetilde{x}_k, log_4, s\_keys, i_0, \widetilde{i}_k, i)) \wedge\\ &\quad \mathbf{event}(\mathsf{ServerAccept}_k(x, \widetilde{x}_k, log_4, s\_keys', i_0', \widetilde{i}_k', i')) \Longrightarrow (i_0, \widetilde{i}_k, i) = (i_0', \widetilde{i}_k', i')\end{aligned} \tag{43}$$

where $\widetilde{i}_k$ contains as many replication indices as the length of k.

In general, $\mathsf{AddReplSid}$ can add only the session identifier to the events, and not the replication indices, because the replication indices are renumbered during the composition. (With the notations of Theorem 2, the indices of Q_{2B} are renumbered.) That would break some correspondences. In the particular case of (31), we could actually add the indices because either the indices of both occurrences of $\mathsf{ServerAccept}_k$ are not renumbered or they are renumbered in the same way, so the correspondence is preserved.

Formally, we do not need (43), because in all our compositions, we use as key exchange protocol (system S_1 in Theorem 2) a protocol that performs a single handshake, so (31) itself is enough. We may still want to prove (43). We can show it by combining several correspondences: in the composed protocol, the event $\mathsf{ServerAccept}_k$ is always preceded by the events $\mathsf{ServerAccept}$ (for the initial handshake) and $\mathsf{ServerAccept}_{k'}$ for each strict prefix k' of k (for the previous handshakes with pre-shared key). These events use as log a prefix of the log for $\mathsf{ServerAccept}_k$. Therefore, if we have two executions of $\mathsf{ServerAccept}_k$ with the same log, $\mathsf{ServerAccept}_k(x, \widetilde{x}_k, log_4, s\_keys, i_0, \widetilde{i}_k, i)$ and $\mathsf{ServerAccept}_k(x, \widetilde{x}_k, log_4, s\_keys', i'_0, \widetilde{i}'_k, i')$, then we have two executions of $\mathsf{ServerAccept}$ with the same log, which implies that they have the same replication index by (31), so $i_0 = i'_0$. For each strict prefix k' of k, we also have two executions of $\mathsf{ServerAccept}_{k'}$ with the same log, which implies that they have the same final replication index by (31) with additional index k' and additional arguments $x, \widetilde{x}_{k'}$, so $\widetilde{i}_k$ and $\widetilde{i}'_k$ have the same $(|k'| + 1)$-th element, where $|k'|$ is the length of k'. By (42), we have $i = i'$. So by combining these results, we have (43).

The same remark applies to the correspondences (33) and (35) as well.

RESEARCH CENTRE
PARIS

2 rue Simone Iff
CS 42112
75589 Paris Cedex 12

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399