# Stateful Protocol Composition (Extended Version)

## DTU Compute Technical Report-2018-03. ISSN: 1601-2321

Andreas V. Hess[1], Sebastian A. Mödersheim[1], and Achim D. Brucker[2]

[1] DTU Compute, Technical University of Denmark, Lyngby, Denmark
{avhe,samo}@dtu.dk
[2] The University of Sheffield, Sheffield, United Kingdom
a.brucker@sheffield.ac.uk

**Abstract.** We prove a parallel compositionality result for protocols with a shared mutable state, i.e., stateful protocols. For protocols satisfying certain compositionality conditions our result shows that analyzing the component protocols in isolation is sufficient to prove security of the more complex composition. Our main contribution is an extension of the compositionality paradigm to stateful protocols where participants maintain shared databases. We also support declassification of shared secrets. As a final contribution we prove the core of our result in Isabelle/HOL, providing a strong correctness guarantee of our proofs.

## 1 Introduction

The typical use of communication networks like the Internet is to run a wide variety of security protocols in parallel, for example TLS, IPSec, DNSSEC, and many others. While the security properties of many of these protocols have been analyzed in great detail, much less research has been devoted to their parallel composition. It is far from self-evident that the parallel composition of secure protocols is still secure, in fact one can systematically construct counterexamples. One such problem is if protocols have similar message structures of different meaning, so that an attacker may be able to abuse messages, or parts thereof, that he has learned in the context of one protocol, and use them in the context of another where the same structure has a different meaning. Thus, we have to exclude that the protocols in some sense "interfere" with each other. However, it is unreasonable to require that the developers of the different protocols have to work together and synchronize with each other. Similarly, we do not want to reason about the composition of several protocols as a whole, neither in manual nor automated verification. Instead, we want a set of sufficient conditions and a composition theorem of the form: every set of protocols that satisfies the conditions yields a secure composition, provided that each protocol is secure in isolation. The conditions should be realistic so that many existing protocols like TLS (without modifications) actually satisfy them, and they should be simple, in the sense that checking them is a static task that does not involve considering the reachable states.

The main contribution of this paper is the extension of the compositionality paradigm to *stateful* protocols, where participants may maintain a database (e.g., a list of valid public keys) independent of sessions. Such databases do not necessarily grow monotonically during protocol execution—we allow, for instance, negative membership checks and deletion of elements from databases. Moreover, we allow for such databases to be *shared* between the protocols to be composed. For instance, in the example of public keys, there could be several different protocols for registering, certifying, and revoking keys that all work on the same public-key database. Since such a shared database can potentially be exploited by the intruder to trigger harmful interferences, an important part of our result is a clear coordination of the ways in which each protocol is allowed to access the database. This coordination is based on assumptions and guarantees on the transactions affecting the database. Moreover, this also allows us to support protocols with the declassification of long-term secrets (e.g., that the private key to a revoked public key may be learned by the intruder without breaking the security goals). The result is so general that it actually also covers many forms of *sequential composition* as a special case, since one can for instance model that one protocol inserts keys into a database of fresh session keys, and another protocol "consumes" and uses them.

The proof of the main result is by a reduction to a problem finding solutions for intruder constraints: given a satisfiable constraint representing an attack on the composition, we show that the projection of the constraints to the individual protocols are satisfiable. This particular tricky part of the proof has been formalized in the interactive theorem prover Isabelle/HOL. This formalization, along with all proofs, is available at `https://people.compute.dtu.dk/samo/composec.html`. An extended version of this paper is also available at this website. Formulation of the problem over intruder constraints allows us to apply our result with a variety of protocol formalisms such as applied-$\pi$ calculus and multi-set rewriting.

The rest of the paper is organized as follows. Preliminaries are introduced in section 2. In section 3 we define stateful constraints and protocols. Afterwards we define protocol composition and introduce a keyserver protocol example in section 4. We define our compositionality conditions and prove our main result in section 5. Finally, we conclude in section 6 and discuss related work. An explanation of the proofs of our technical results can be found in appendix A.

## 2 Preliminaries

### 2.1 Terms and Substitutions

We model terms over a countable signature $\Sigma$ of function symbols and a countably infinite set $\mathcal{V}$ of variable symbols. We do not fix here a particular set of cryptographic operators but rather parameterize our theory over arbitrary $\Sigma$.

A term is either a variable $x \in \mathcal{V}$ or a composed term of the form $f(t_1, \ldots, t_n)$ where $f \in \Sigma^n$ and $t_i$ are terms and $\Sigma^n$ denotes the symbols in $\Sigma$ of *arity* $n$. The set of *constants* $\mathcal{C}$ is defined as $\Sigma^0$. The set of variables of a term $t$ is denoted by $fv(t)$ and if $fv(t) = \emptyset$ then $t$ is *ground*. Both of these notions are extended to sets of terms. By $\sqsubseteq$ we denote the *subterm* relation.

Substitutions are defined as functions from variables to terms. The domain of a substitution $\delta$ is denoted by $dom(\delta)$ and is defined as the set of variables that are not mapped to themselves by $\delta$: $dom(\delta) \equiv \{x \in \mathcal{V} \mid \delta(x) \neq x\}$. The substitution image, $img(\delta)$, is then defined as the image of $dom(\delta)$ under $\delta$: $img(\delta) \equiv \delta(dom(\delta))$. If the image of $\delta$ is ground then $\delta$ is said to be a *ground substitution*. Additionally, we define an *interpretation* to be a substitution that assigns a ground term to every variable: $\mathcal{I}$ is an interpretation iff $dom(\mathcal{I}) = \mathcal{V}$ and $img(\mathcal{I})$ is ground. We extend substitutions to functions on terms and set of terms as expected. For substitutions $\delta$ with finite domain we will usually use the common value mapping notation: $\delta = [x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$. Finally, a substitution $\delta$ is a *unifier of* terms $t$ and $t'$ iff $\delta(t) = \delta(t')$.

### 2.2 The Intruder Model

The intruder model follows the standard of Dolev and Yao, roughly, the intruder can encrypt and decrypt terms where he has the respective keys, but he cannot break the cryptography. This is often done by a set of rules specialized to the concrete cryptographic functions, but since our model is parameterized over an arbitrary set $\Sigma$, we also need to parameterize it over (a) a predicate `public` over $\Sigma$ that says for each function whether it is available to the intruder and (b) a function $\mathsf{Ana}$ that takes a term $t$ and returns a pair $(K, T)$ of sets of terms. The meaning is: from the term $t$ the intruder can obtain the terms $T$, provided that he knows all the "keys" in the set $K$. For instance if $\mathsf{crypt}$ is a public function symbol to represent asymmetric encryption and $\mathsf{inv}$ is a private function symbol (i.e., $\neg\texttt{public}(\mathsf{inv})$) mapping public keys to the corresponding private key, then we may define $\mathsf{Ana}(\mathsf{crypt}(k, m)) = (\{\mathsf{inv}(k)\}, \{m\})$ for any terms $k$ and $m$. Thus we can inductively define the relation $\vdash$, where $M \vdash t$ means that an intruder who knows the set of terms $M$ can derive the message $t$ as the least relation that includes $M$, is closed under composition with public functions and is closed under analysis with $\mathsf{Ana}$ as follows where $\Sigma^n_{pub} \equiv \{f \in \Sigma^n \mid \texttt{public}(f)\}$:

**Definition 1 (Intruder model).**

$$\frac{}{M \vdash t}\ \begin{array}{c}(\mathit{Axiom}),\\ t \in M\end{array} \qquad \frac{M \vdash t_1 \quad \cdots \quad M \vdash t_n}{M \vdash f(t_1, \ldots, t_n)}\ \begin{array}{c}(\mathit{Compose}),\\ f \in \Sigma^n_{pub}\end{array}$$

$$\frac{M \vdash t \quad M \vdash k_1 \quad \cdots \quad M \vdash k_n}{M \vdash t_i}\ \begin{array}{c}(\mathit{Decompose}), \mathsf{Ana}(t) = (K, T),\\ t_i \in T, K = \{k_1, \ldots, k_n\}\end{array}$$

Note that [15] in contrast considers only public function symbols; one can simulate however a private function symbol of arity $n$ by a public function symbol of arity $n+1$ where the additional argument is used with a special constant that is never given to the intruder; in this way all results can be lifted to a model with both private and public function symbols. For instance we can encode $\mathsf{inv} \in \Sigma^1$ in terms of a public symbol $\mathsf{inv}' \in \Sigma^2$ and a special secret constant $\mathsf{sec}_{\mathsf{inv}}$.

Our results will not work with an arbitrary analysis function, so we make the following requirements on $\mathsf{Ana}$:

1. $\mathsf{Ana}(x) = (\emptyset, \emptyset)$ for variables $x \in \mathcal{V}$,
2. $\mathsf{Ana}(f(t_1, \dots, t_n)) = (K, T)$ implies $T \subseteq \{t_1, \dots, t_n\}$, finite $K$, and $fv(K) \subseteq fv(f(t_1, \dots, t_n))$,
3. $\mathsf{Ana}(f(t_1, \dots, t_n)) = (K, T)$ implies $\mathsf{Ana}(\delta(f(t_1, \dots, t_n))) = (\delta(K), \delta(T))$.

Note that $\mathsf{Ana}$ must be defined for arbitrary terms, including terms with variables (while the standard Dolev-Yao deduction is typically applied to ground terms). The three conditions regulate that $\mathsf{Ana}$ is also meaningful on symbolic terms. The first requirement says that we cannot analyze a variable. The second requirement says that the result of the analysis are *immediate* subterms of the term being analyzed, and the keys can be any finite set of terms, but built with only variables that occur in the term being analyzed. The third requirement says that analysis does not change its behavior when instantiating a term (that is not a variable).

Our requirements on $\mathsf{Ana}$ are a bit simpler than the ones in [15]. There, also the key-terms $K$ have to be subterms of the analyzed term, while the third requirement is stated only for terms that do not yield $(\emptyset, \emptyset)$. While this is more relaxed, it is a quite roundabout condition that was introduced to handle a model of public-key encryption where public keys were modeled with a function $\mathsf{pub}$ from private to public keys. Since we allow also for private functions, and since we have less restrictions on the key-terms $K$ of $\mathsf{Ana}$, we can also work with the private function $\mathsf{inv}$ from public to private keys instead, and do not need this special case. Since it simplifies many things, we decided to stick with it, but note that our results would also work similarly with the definition from [15].

*Example 1.* We model asymmetric encryption and signatures with the following $\mathsf{Ana}$ theory: $\mathsf{Ana}(\mathsf{crypt}(k, m)) = (\{\mathsf{inv}(k)\}, \{m\})$, $\mathsf{Ana}(\mathsf{sign}(k, m)) = (\emptyset, \{m\})$. We will also later use some transparent functions: $\mathsf{Ana}(\mathsf{pair}(t, t')) = (\emptyset, \{t, t'\})$ and $\mathsf{Ana}(\mathsf{update}(s, t, u, v)) = (\emptyset, \{s, t, u, v\})$. For all other terms $t$: $\mathsf{Ana}(t) = (\emptyset, \emptyset)$.

## 3 Stateful Protocols

We now introduce a strand-based protocol formalism for stateful protocols adapted from [16]. This formalism is compact and reduced to the key concepts needed here, while more complex formalisms like process calculi can easily be fitted similarly. The semantics is defined by a symbolic transition system where constraints are built-up during transitions. The models of the constraints then constitute the concrete protocol runs. We will use a typing result that shows that for a large class of protocols, it is without loss of attacks to restrict the constraints to well-typed models [16]. A minor contribution of the present paper (besides small adaptions) is the proof of this result in Isabelle/HOL.

### 3.1 Stateful Symbolic Constraints

We use *intruder constraints* as a key concept for reasoning about protocol executions and attacks. This is in fact applicable with a variety of protocol verification formalisms, such as process calculi or multi-set rewrite rules. The idea is to define a *symbolic* transition system where the variables of sent and received messages of the original protocol formalism are not instantiated (only renamed as necessary) and formulate symbolic constraints on these variables: the intruder needs to be able to construct each message an honest agent receives from the messages the honest agents have sent up to that point. When equipping these constraints also with equalities and inequalities, the set of all executions (and the attack predicates) of many formalisms like Applied $\pi$-calculus can be described by a set of constraints. An attack can then be defined by satisfiability of a constraint in which the intruder produces a secret. *Stateful constraints* can furthermore express queries and updates on databases. They are defined as finite sequences of *steps* and are built from the following grammar where $t$ and $t'$ ranges over terms and $\bar{x}$ over finite variable sequences $x_1, \dots, x_n$:

$$\begin{aligned}\mathcal{A} ::= {} & \mathsf{send}(t).\mathcal{A} \mid \mathsf{receive}(t).\mathcal{A} \mid t \doteq t'.\mathcal{A} \mid (\forall \bar{x}.\ t \mathrel{\dot{\neq}} t').\mathcal{A} \mid \\ & \mathsf{insert}(t, t').\mathcal{A} \mid \mathsf{delete}(t, t').\mathcal{A} \mid t \mathrel{\dot{\in}} t'.\mathcal{A} \mid (\forall \bar{x}.\ t \mathrel{\dot{\notin}} t').\mathcal{A} \mid 0\end{aligned}$$

Instead of $\forall \bar{x}.\ t \mathrel{\dot{\neq}} t'$ and $\forall \bar{x}.\ t \mathrel{\dot{\notin}} t'$ we may write $t \mathrel{\dot{\neq}} t'$ and $t \mathrel{\dot{\notin}} t'$ whenever $\bar{x}$ is the empty sequence. We may also write $t \mathrel{\dot{\notin}} f(\_)$ for $f \in \Sigma^n$ as an abbreviation of $\forall x_1, \dots, x_n.\ t \mathrel{\dot{\notin}} f(x_1, \dots, x_n)$. The *bound variables* of a constraint $\mathcal{A}$ consists of its variable sequences while the remaining variables, $fv(\mathcal{A})$, are the *free variables.* Also, by $trms(\mathcal{A})$ we denote the set of terms occurring in $\mathcal{A}$ and the *set of set operations* of $\mathcal{A}$, called $setops(\mathcal{A})$, is defined as follows where $(\cdot, \cdot) \in \Sigma^2_{pub}$:

$$setops(\mathcal{A}) \equiv \{(t, s) \mid \mathsf{insert}(t, s) \text{ or } \mathsf{delete}(t, s) \text{ or } t \mathrel{\dot{\in}} s \text{ or } \forall \bar{x}.\ t \mathrel{\dot{\notin}} s \text{ occurs in } \mathcal{A}\}$$

For the semantics of constraints we first define a predicate $[\![M, D; \mathcal{A}]\!]\ \mathcal{I}$, where $M$ is a ground set of terms (the intruder knowledge), $D$ is a ground set of tuples (the state of the sets), $\mathcal{A}$ is a constraint, and $\mathcal{I}$ is an interpretation as follows:

$$\begin{array}{rcl}
[\![M, D; 0]\!]\ \mathcal{I} & \text{iff} & \mathit{true} \\
[\![M, D; \mathsf{send}(t).\mathcal{A}]\!]\ \mathcal{I} & \text{iff} & M \vdash \mathcal{I}(t) \text{ and } [\![M, D; \mathcal{A}]\!]\ \mathcal{I} \\
[\![M, D; \mathsf{receive}(t).\mathcal{A}]\!]\ \mathcal{I} & \text{iff} & [\![\{\mathcal{I}(t)\} \cup M, D; \mathcal{A}]\!]\ \mathcal{I} \\
[\![M, D; t \doteq t'.\mathcal{A}]\!]\ \mathcal{I} & \text{iff} & \mathcal{I}(t) = \mathcal{I}(t') \text{ and } [\![M, D; \mathcal{A}]\!]\ \mathcal{I} \\
[\![M, D; (\forall \bar{x}.\ t \mathrel{\dot{\neq}} t').\mathcal{A}]\!]\ \mathcal{I} & \text{iff} & [\![M, D; \mathcal{A}]\!]\ \mathcal{I} \text{ and } \mathcal{I}(\delta(t)) \neq \mathcal{I}(\delta(t')) \\
 & & \text{for all ground substitutions } \delta \text{ with domain } \bar{x} \\
[\![M, D; \mathsf{insert}(t, s).\mathcal{A}]\!]\ \mathcal{I} & \text{iff} & [\![M, \{\mathcal{I}((t, s))\} \cup D; \mathcal{A}]\!]\ \mathcal{I} \\
[\![M, D; \mathsf{delete}(t, s).\mathcal{A}]\!]\ \mathcal{I} & \text{iff} & [\![M, D \setminus \{\mathcal{I}((t, s))\}; \mathcal{A}]\!]\ \mathcal{I} \\
[\![M, D; t \mathrel{\dot{\in}} s.\mathcal{A}]\!]\ \mathcal{I} & \text{iff} & \mathcal{I}((t, s)) \in D \text{ and } [\![M, D; \mathcal{A}]\!]\ \mathcal{I} \\
[\![M, D; (\forall \bar{x}.\ t \mathrel{\dot{\notin}} s).\mathcal{A}]\!]\ \mathcal{I} & \text{iff} & [\![M, D; \mathcal{A}]\!]\ \mathcal{I} \text{ and } \mathcal{I}(\delta((t, s))) \notin D \\
 & & \text{for all ground substitutions } \delta \text{ with domain } \bar{x}
\end{array}$$

We then define that $\mathcal{I}$ is a *model* of $\mathcal{A}$, written $\mathcal{I} \models \mathcal{A}$, iff $[\![\emptyset, \emptyset; \mathcal{A}]\!]\ \mathcal{I}$.

A crucial requirement on constraints is that they are well-formed in the sense that every variable first occurs in a message the intruder sends, or in a positive check like $t \doteq t'$ or $t \mathrel{\dot{\in}} s$, and that the intruder knowledge monotonically grows over time. The latter condition is already built-in in our constraint notation, the former is expressed as follows: A constraint $\mathcal{A}$ is *well-formed w.r.t.* the set of variables $X$ (or just *well-formed* if $X = \emptyset$) iff the free variables and the bound variables of $\mathcal{A}$ are disjoint and $\mathit{wf}_X(\mathcal{A})$ holds where:

$$\begin{array}{lll}
\mathit{wf}_X(0) & \text{iff} & \mathit{true} \\
\mathit{wf}_X(\mathsf{receive}(t).\mathcal{A}) & \text{iff} & \mathit{fv}(t) \subseteq X \text{ and } \mathit{wf}_X(\mathcal{A}) \\
\mathit{wf}_X(\mathsf{send}(t).\mathcal{A}) & \text{iff} & \mathit{wf}_{X \cup \mathit{fv}(t)}(\mathcal{A}) \\
\mathit{wf}_X(t \doteq t'.\mathcal{A}) & \text{iff} & \mathit{fv}(t') \subseteq X \text{ and } \mathit{wf}_{X \cup \mathit{fv}(t)}(\mathcal{A}) \\
\mathit{wf}_X(\mathsf{insert}(t, t').\mathcal{A}) & \text{iff} & \mathit{fv}(t) \cup \mathit{fv}(t') \subseteq X \text{ and } \mathit{wf}_X(\mathcal{A}) \\
\mathit{wf}_X(\mathsf{delete}(t, t').\mathcal{A}) & \text{iff} & \mathit{fv}(t) \cup \mathit{fv}(t') \subseteq X \text{ and } \mathit{wf}_X(\mathcal{A}) \\
\mathit{wf}_X(t \mathrel{\dot{\in}} t'.\mathcal{A}) & \text{iff} & \mathit{wf}_{X \cup \mathit{fv}(t) \cup \mathit{fv}(t')}(\mathcal{A}) \\
\mathit{wf}_X(\mathfrak{a}.\mathcal{A}) & \text{iff} & \mathit{wf}_X(\mathcal{A}) \text{ otherwise}
\end{array}$$

Note that this allows to "introduce" variables in a send step, on the left-hand side of an equation, or in a positive set-membership check (and we will work only with well-formed constraints throughout the paper).

### 3.2 Typed Model

Our result is based on a typed model of protocols, i.e., where the intruder by definition cannot send ill-typed messages. [16] shows that this is not a restriction for a large class of so-called *type-flaw resistant* stateful protocols, since for every ill-typed attack also exists a well-typed one. This gives a sufficient condition for protocols to satisfy a prerequisite of our compositionality result. The definition of typed model is then as follows. Type expressions are terms built over the function symbols of $\Sigma$ and a finite set $\mathfrak{T}_a$ of *atomic* types like $\mathsf{Agent}$ and $\mathsf{Nonce}$. Further, we define a typing function $\Gamma$ that assigns to every variable a type, to every constant an atomic type, and that is extended to composed terms as follows: $\Gamma(f(t_1, \dots, t_n)) = f(\Gamma(t_1), \dots, \Gamma(t_n))$ for every $f \in \Sigma^n \setminus \mathcal{C}$ and terms $t_i$. We also require that $\{c \in \mathcal{C} \mid \mathtt{public}(c), \Gamma(c) = \beta\}$ is infinite for each $\beta \in \mathfrak{T}_a$, thus giving the intruder access to an infinite supply of terms of each atomic type.

The sufficient condition for a protocol to satisfy the typing result is now based on the following notions. A substitution $\delta$ is *well-typed* iff $\Gamma(x) = \Gamma(\delta(x))$ for all $x \in \mathcal{V}$. Given a set of messages that occur in a protocol we define the following set of sub-message patterns, intuitively the ones that may occur during constraint reduction:

**Definition 2 (Sub-message patterns).** *The* sub-message patterns $\mathit{SMP}(M)$ *for a set of messages $M$ is defined as the least set satisfying the following rules:*

1. *$M \subseteq \mathit{SMP}(M)$.*
2. *If $t \in \mathit{SMP}(M)$ and $t' \sqsubseteq t$ then $t' \in \mathit{SMP}(M)$.*
3. *If $t \in \mathit{SMP}(M)$ and $\delta$ is a well-typed substitution then $\delta(t) \in \mathit{SMP}(M)$.*

*4. If $t \in SMP(M)$ and $\mathsf{Ana}(t) = (K, T)$ then $K \subseteq SMP(M)$.*

The sufficient condition for the typing result is now that non-variable submessage patterns have no unifier unless they have the same type:

**Definition 3 (Type-flaw resistance).** *We say a set $M$ of messages is* type-flaw resistant *iff $\forall t, t' \in SMP(M) \setminus \mathcal{V}.\ (\exists \delta.\ \delta(t) = \delta(t')) \longrightarrow \Gamma(t) = \Gamma(t')$. We may also apply the notion of type-flaw resistance to a constraint $\mathcal{A}$ to mean that:*

- *$trms(\mathcal{A}) \cup setops(\mathcal{A})$ is type-flaw resistant,*
- *if $t$ and $t'$ are unifiable then $\Gamma(t) = \Gamma(t')$, for all $t \doteq t'$ occurring in $\mathcal{A}$,*
- *$\Gamma(fv(t) \cup fv(t')) \subseteq \mathfrak{T}_a$ for all $\mathsf{insert}(t, t')$ and $\mathsf{delete}(t, t')$ occurring in $\mathcal{A}$, and*
- *$\Gamma((fv(t) \cup fv(t')) \setminus \bar{x}) \subseteq \mathfrak{T}_a$ for all $\forall \bar{x}.\ t \not\doteq t'$ and $\forall \bar{x}.\ t \not\dot\in t'$ occurring in $\mathcal{A}$.*

Compared to [16] we have slightly more relaxed requirements on variables: we allow variables of composed type in $t \dot\in s$ checks and in bound variables.

We have formalized in Isabelle/HOL the following typing result theorem from [16], which shows that for type-flaw resistant protocols it is safe to check satisfiability of constraints within the typed model:

**Theorem 1** *If $\mathcal{A}$ is a well-formed, type-flaw resistant constraint, and if $\mathcal{I} \models \mathcal{A}$, then there exists a well-typed interpretation $\mathcal{I}_\tau$ such that $\mathcal{I}_\tau \models \mathcal{A}$.*

### 3.3 Protocol Semantics

Protocols are defined as sets $\mathcal{P} = \{R_1, \ldots\}$ of *transaction rules* of the form: $R_i = \forall x_1 \in T_1, \ldots, x_n \in T_n.\ \mathsf{new}\ y_1, \ldots, y_m.\ S$ where $S$ is a *transaction strand*, i.e., of the form $\mathsf{receive}(t_1).\cdots.\mathsf{receive}(t_k).\phi_1 \cdots .\phi_{k'}.\mathsf{send}(t'_1).\cdots.\mathsf{send}(t'_{k''})$ where

$$\phi ::= t \doteq t' \mid \forall \bar{x}.\ t \not\doteq t' \mid t \dot\in t' \mid \forall \bar{x}.\ t \not\dot\in t' \mid \mathsf{insert}(t, t') \mid \mathsf{delete}(t, t')$$

The prefix $\forall x_1 \in T_1, \ldots, x_n \in T_n$ denotes that the transaction strand $S$ is applicable for instantiations $\sigma$ of the $x_i$ variables where $\sigma(x_i) \in T_i$. The construct $\mathsf{new}\ y, \ldots, y_m$ represents that the occurrences of the variables $y_i$ in the transaction strand $S$ will be instantiated with fresh terms. We extend $trms(\cdot)$ and $setops(\cdot)$ to transactions strands, rules, and protocols as expected.

We define a transition relation $\Rightarrow^\bullet_{\mathcal{P}}$ for protocol $\mathcal{P}$ where states are constraints and the initial state is the empty constraint 0. First we define the *dual* of a transaction strand $S$, written $dual(S)$, as "swapping" the direction of the sent and received messages of $S$: $dual(\mathsf{send}(t).S) = \mathsf{receive}(t).dual(S)$, $dual(\mathsf{receive}(t).S) = \mathsf{send}(t).dual(S)$, and otherwise $dual(\mathfrak{s}.S) = \mathfrak{s}.dual(S)$. The transition $\mathcal{A} \Rightarrow^\bullet_{\mathcal{P}} \mathcal{A}.dual(\alpha(\sigma(S)))$ is then applicable if these conditions are met:

1. $(\forall x_1 \in T_1, \ldots, x_n \in T_n.\ \mathsf{new}\ y_1, \ldots, y_m.\ S) \in \mathcal{P}$,
2. $dom(\sigma) = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$,
3. $\sigma(x_i) \in T_i$ for all $i \in \{1, \ldots, n\}$,
4. $\sigma(y_i)$ is a fresh ground term of type $\Gamma(y_i)$ for all $i \in \{1, \ldots, m\}$, and
5. $\alpha$ is a variable-renaming of the variables of $\sigma(S)$ where $\alpha$ is well-typed and the variables in $img(\alpha)$ do not occur in $\sigma(S)$.

Hence transaction rules are processed atomically, and converted into constraints, during transitions. Note that each transaction rule can be executed arbitrarily often and so we support an unbounded number of “sessions”. For instance, the transaction rule $\forall A \in \mathsf{Hon}.\ \mathsf{new}\ PK.\ \mathsf{insert}(PK, \mathsf{ring}(A))$ models that each honest agent $a \in \mathsf{Hon}$ can insert one fresh key into its keyring $\mathsf{ring}(a)$ during each application of the transaction rule. This rule can be executed any number of times with any agent $a \in \mathsf{Hon}$ and a fresh value for $PK$ each time.

We say that a constraint $\mathcal{A}$ is *reachable* in protocol $\mathcal{P}$ if $0 \Rightarrow^{\bullet\star}_{\mathcal{P}} \mathcal{A}$ where $\Rightarrow^{\bullet\star}_{\mathcal{P}}$ denotes the transitive reflexive closure of $\Rightarrow^{\bullet}_{\mathcal{P}}$. We need to ensure that these constraints are well-formed and we will therefore always assume the following sufficient requirement on the protocols $\mathcal{P}$ that we work with: for any transaction strand $S$ occurring in any rule $\forall x_1 \in T_1, \dots, x_n \in T_n.\ \mathsf{new}\ y_1, \dots, y_m.\ S$ of $\mathcal{P}$ the constraint $\mathit{dual}(S)$ is well-formed w.r.t. the variables $\{x_1, \dots, x_n, y_1, \dots, y_m\}$. In other words, the variables of $S$ must first occur in either a $\mathsf{receive}$ step, a positive check ($\doteq$, $\dot{\in}$), or be part of $\{x_1, \dots, x_n, y_1, \dots, y_m\}$.

To model goal violations of a protocol $\mathcal{P}$ we first fix a special constant unique to $\mathcal{P}$, e.g., $\mathsf{attack}_{\mathcal{P}}$. Secondly, we add the rule $\mathsf{receive}(\mathsf{attack}_{\mathcal{P}})$ to $\mathcal{P}$ that we use as a signal for when an attack has occurred. The protocol then has a (well-typed) attack if there exists a (well-typed) satisfiable reachable constraint of the form $\mathcal{A}.\mathsf{send}(\mathsf{attack}_{\mathcal{P}})$. A protocol with no attacks is *secure*.

## 4 Composition and a Running Example

The core definition of this paper is rather simple: we define the *parallel composition* $\mathcal{P}_1 \parallel \mathcal{P}_2$ of protocols $\mathcal{P}_1$ and $\mathcal{P}_2$ as their union: $\mathcal{P}_1 \parallel \mathcal{P}_2 \equiv \mathcal{P}_1 \cup \mathcal{P}_2$. Protocols $\mathcal{P}_1$ and $\mathcal{P}_2$ are also referred to as the *component protocols* of the composition $\mathcal{P}_1 \parallel \mathcal{P}_2$. For such a composed protocol the reachable constraints in $\mathcal{P}_1 \parallel \mathcal{P}_2$ will in general contain steps originating from both component protocols. To keep track of where a step in a constraint originated we assign to each step a *label* $\ell \in \{1, 2, \star\}$. The steps that are exclusive to the first component are marked with 1 while the steps exclusive to the second are marked with 2. In addition to the protocol-specific labels we also have a special label $\star$ that we explain later.

Let $\mathcal{A}$ be a constraint with labels and $\ell \in \{1, 2, \star\}$, we define $\mathcal{A}|_\ell$ to be the projection of $\mathcal{A}$ to the steps labeled $\ell$ or $\star$ (so the $\star$-steps are kept in every a projection). We extend projections to transaction rules and protocols as expected. We may also write $\mathcal{P}^\star$ instead of $\mathcal{P}|_\star$.

### 4.1 A Keyserver Example

As a running example, we define two keyserver protocols that share the same database of valid public keys registered at the keyserver. In a nutshell, the first protocol $\mathcal{P}_{ks,1} = \{1\colon \mathsf{receive}(\mathsf{attack}_1), R_1^1, \dots, R_1^9\}$ allows users to register public keys out of band and to updating an existing key with a new one (revoking the old key), while the second protocol $\mathcal{P}_{ks,2} = \{2\colon \mathsf{receive}(\mathsf{attack}_2), R_2^1, \dots, R_2^9\}$ uses a different mechanism to register new public keys. Thus, both protocols use the shared database $\mathsf{valid}(A, S)$ (where $A$ is the name of the user and $S$ the name of the server).

There are three atomic types used in the example: the type of agents $\mathsf{Agent}$, public keys $\mathsf{PubKey}$, and the type $\mathsf{Attack}$ of the $\mathsf{attack}_i$ constants. We partition the constants of type $\mathsf{Agent}$ into the honest users $\mathsf{Hon}$, the dishonest users $\mathsf{Dis}$, and the keyservers $\mathsf{Ser}$. There are sets for authentication goals $\mathsf{begin}_1$, $\mathsf{end}_1$, $\mathsf{begin}_2$, and $\mathsf{begin}_2$, and all protocol steps related to these sets are highlighted in blue; let us first ignore these.

*Protocol $\mathcal{P}_{ks,1}$* In the first protocol, the following rule models that an honest user registers a new public key $PK$ out of band (e.g., by physically visiting a registration site):

$$\begin{array}{l} \forall A \in \mathsf{Hon}, S \in \mathsf{Ser}.\ \mathsf{new}\ PK. \\ \quad 1\colon \mathsf{insert}(PK, \mathsf{ring}(A)). \\ \quad \star\colon \mathsf{insert}(PK, \mathsf{valid}(A, S)). \\ \quad \star\colon \mathsf{insert}(PK, \mathsf{begin}_1(A, S)). \\ \quad \star\colon \mathsf{insert}(PK, \mathsf{end}_1(A, S)). \\ \quad \star\colon \mathsf{send}(PK) \end{array} \tag{$R_1^5$}$$

This is achieved by inserting $PK$ (in the same transaction) both into the users keyring $\mathsf{ring}(A)$ and into the shared database $\mathsf{valid}(A, S)$. There is also a corresponding rule for dishonest users:

$$\begin{array}{l} \forall A \in \mathsf{Dis}, S \in \mathsf{Ser}. \\ \quad 1\colon \mathsf{receive}(PK). \\ \quad \star\colon \forall A', S'.\ PK\ \dot{\notin}\ \mathsf{valid}(A', S'). \\ \quad \star\colon \mathsf{insert}(PK, \mathsf{valid}(A, S)) \end{array} \tag{$R_1^9$}$$

Note that dishonest users may register in their name any key they know (hence the $\mathsf{receive}(PK)$ step), so the key is not necessarily freshly created; also we do not model a key ring for them. Rule $R_1^4$ gives the intruder arbitrarily many fresh key pairs:

$$\begin{array}{l} \forall A \in \mathsf{Dis}.\ \mathsf{new}\ PK. \\ \quad \star\colon \mathsf{send}(PK). \\ \quad \star\colon \mathsf{send}(\mathsf{inv}(PK)) \end{array} \tag{$R_1^4$}$$

Secondly, we model a key update with revocation of old keys. To request an update of key $PK$ with a newly generated key $NPK$ at server $S$, an honest user sends $NPK$ signed with $PK$ as in the following rule:

$$\begin{array}{l} \forall A \in \mathsf{Hon}, S \in \mathsf{Ser}.\ \mathsf{new}\ NPK. \\ \quad 1\colon PK\ \dot{\in}\ \mathsf{ring}(A). \\ \quad 1\colon \mathsf{delete}(PK, \mathsf{ring}(A)). \\ \quad 1\colon \mathsf{insert}(NPK, \mathsf{ring}(A)). \\ \quad \star\colon \mathsf{insert}(NPK, \mathsf{begin}_1(A, S)). \\ \quad \star\colon \mathsf{send}(NPK). \\ \quad 1\colon \mathsf{send}(\mathsf{sign}(\mathsf{inv}(PK), \mathsf{pair}(A, NPK))) \end{array} \tag{$R_1^6$}$$

(For this rule, there is no equivalent for the dishonest agents, since they may produce an arbitrary update request message.)

The rule $R_1^7$ shows how $S$ receives the update message from an honest agent:

$$\begin{array}{l}
\forall A \in \mathsf{Hon}, S \in \mathsf{Ser}.\\
\quad 1\colon \mathsf{receive}(\mathsf{sign}(\mathsf{inv}(PK), \mathsf{pair}(A, NPK))).\\
\quad \star\colon PK \mathrel{\dot\in} \mathsf{valid}(A, S).\\
\quad \star\colon \forall A', S'.\ NPK \mathrel{\dot\notin} \mathsf{valid}(A', S').\\
\quad 1\colon \forall A', S'.\ NPK \mathrel{\dot\notin} \mathsf{revoked}(A', S').\\
\quad \star\colon NPK \mathrel{\dot\in} \mathsf{begin}_1(A, S).\\
\quad \star\colon NPK \mathrel{\dot\notin} \mathsf{end}_1(A, S).\\
\quad \star\colon \mathsf{delete}(PK, \mathsf{valid}(A, S)).\\
\quad \star\colon \mathsf{insert}(NPK, \mathsf{valid}(A, S)).\\
\quad 1\colon \mathsf{insert}(PK, \mathsf{revoked}(A, S)).\\
\quad \star\colon \mathsf{insert}(NPK, \mathsf{end}_1(A, S)).\\
\quad \star\colon \mathsf{send}(\mathsf{inv}(PK))
\end{array} \tag{$R_1^7$}$$

It first checks that the key $PK$ is currently valid, and that $NPK$ is neither registered as valid or revoked. If all checks succeed, updates its databases accordingly. Also, we reveal here $\mathsf{inv}(PK)$, in order to specify that the protocol must even be secure when old private keys are leaked. This is also our example for declassification of a secret that is shared between two protocols. The rule $R_1^8$ is the pendant for dishonest agents:

$$\begin{array}{l}
\forall A \in \mathsf{Dis}, S \in \mathsf{Ser}.\\
\quad 1\colon \mathsf{receive}(\mathsf{sign}(\mathsf{inv}(PK), \mathsf{pair}(A, NPK))).\\
\quad \star\colon PK \mathrel{\dot\in} \mathsf{valid}(A, S).\\
\quad \star\colon \forall A', S'.\ NPK \mathrel{\dot\notin} \mathsf{valid}(A', S').\\
\quad 1\colon \forall A', S'.\ NPK \mathrel{\dot\notin} \mathsf{revoked}(A', S').\\
\quad \star\colon \mathsf{delete}(PK, \mathsf{valid}(A, S)).\\
\quad \star\colon \mathsf{insert}(NPK, \mathsf{valid}(A, S)).\\
\quad 1\colon \mathsf{insert}(PK, \mathsf{revoked}(A, S))
\end{array} \tag{$R_1^8$}$$

*Protocol $\mathcal{P}_{ks,2}$* The second protocol has another mechanism to register new keys: every user has a password $\mathsf{pw}(A, S)$ with the server. The dishonest agents reveal their password to the intruder with rule $R_2^7$):

$$\begin{array}{l}
\forall A \in \mathsf{Dis}, S \in \mathsf{Ser}.\\
\quad 2\colon \mathsf{send}(\mathsf{pw}(A, S))
\end{array} \tag{$R_2^7$}$$

Instead of using a (possibly weak) password for an encryption, the registration message is encrypted with the public key of the server:

$$\begin{array}{l}
\forall A \in \mathsf{Hon}, S \in \mathsf{Ser}.\ \mathsf{new}\ NPK.\\
\quad 2\colon PK \mathrel{\dot\in} \mathsf{pubkeys}(S).\\
\quad \star\colon \mathsf{insert}(NPK, \mathsf{begin}_2(A, S)).\\
\quad \star\colon \mathsf{send}(NPK).\\
\quad 2\colon \mathsf{send}(\mathsf{crypt}(PK, \mathsf{update}(A, S, NPK, \mathsf{pw}(A, S))))
\end{array} \tag{$R_2^5$}$$

For uniformity, we model the server's public keys in a set $\mathsf{pubkeys}(S)$ that is initialized with rule $R_2^9$ (in fact, the server may thus have multiple public keys):

$$\begin{array}{l}\forall S \in \mathsf{Ser}.\ \mathsf{new}\ PK.\\ \quad 2\colon \mathsf{insert}(PK, \mathsf{pubkeys}(S)).\\ \quad \star\colon \mathsf{send}(PK)\end{array} \tag{$R_2^9$}$$

Rule $R_2^6$ models how the server receives a registration request (in case of honest users):

$$\begin{array}{l}\forall A \in \mathsf{Hon}, S \in \mathsf{Ser}.\\ \quad 2\colon \mathsf{receive}(\mathsf{crypt}(PK, \mathsf{update}(A, S, NPK, \mathsf{pw}(A, S)))).\\ \quad 2\colon PK \mathrel{\dot{\in}} \mathsf{pubkeys}(S).\\ \quad 2\colon \forall S'.\ NPK \mathrel{\dot{\notin}} \mathsf{pubkeys}(S').\\ \quad 2\colon \forall A', S'.\ NPK \mathrel{\dot{\notin}} \mathsf{seen}(A', S').\\ \quad \star\colon NPK \mathrel{\dot{\in}} \mathsf{begin}_2(A, S).\\ \quad \star\colon NPK \mathrel{\dot{\notin}} \mathsf{end}_2(A, S).\\ \quad \star\colon \mathsf{insert}(NPK, \mathsf{valid}(A, S)).\\ \quad \star\colon \mathsf{insert}(NPK, \mathsf{end}_2(A, S)).\\ \quad 2\colon \mathsf{insert}(NPK, \mathsf{seen}(A))\end{array} \tag{$R_2^6$}$$

To protect against replay, the server uses a set $\mathsf{seen}$ of seen keys (this may in a real implementation be a buffer-timestamp mechanism). Finally, rule $R_2^8$ is the pendant for the dishonest users:

$$\begin{array}{l}\forall A \in \mathsf{Dis}, S \in \mathsf{Ser}.\\ \quad 2\colon \mathsf{receive}(\mathsf{crypt}(PK, \mathsf{update}(A, S, NPK, \mathsf{pw}(A, S)))).\\ \quad 2\colon PK \mathrel{\dot{\in}} \mathsf{pubkeys}(S).\\ \quad 2\colon \forall S'.\ NPK \mathrel{\dot{\notin}} \mathsf{pubkeys}(S').\\ \quad 2\colon \forall A', S'.\ NPK \mathrel{\dot{\notin}} \mathsf{seen}(A', S').\\ \quad \star\colon \mathsf{insert}(PK, \mathsf{valid}(A, S)).\\ \quad 2\colon \mathsf{insert}(PK, \mathsf{seen}(A))\end{array} \tag{$R_2^8$}$$

As in the first protocol we also have a rule in $\mathcal{P}_{ks,2}$ that gives the intruder arbitrarily many fresh key pairs:

$$\begin{array}{l}\forall A \in \mathsf{Dis}.\ \mathsf{new}\ PK.\\ \quad \star\colon \mathsf{send}(PK).\\ \quad \star\colon \mathsf{send}(\mathsf{inv}(PK))\end{array} \tag{$R_2^4$}$$

*Secrecy* If a valid private key of an honest agent becomes known to the intruder then there is an attack. We define such a secrecy goal as follows, where $i = 1$ in $\mathcal{P}_{ks,1}$ and $i = 2$ in $\mathcal{P}_{ks,2}$:

$$\begin{array}{l}\forall A \in \mathsf{Hon}, S \in \mathsf{Ser}.\\ \quad i\colon \mathsf{receive}(\mathsf{inv}(PK)).\\ \quad \star\colon PK \mathrel{\dot{\in}} \mathsf{valid}(A, S).\\ \quad i\colon \mathsf{send}(\mathsf{attack}_i)\end{array} \tag{$R_i^1$}$$

*Authentication* Besides the secrecy goal $R_i^1$ that no valid private key of an honest agent may ever be known by the intruder, the crucial authentication goal is that all insertions into $\mathsf{valid}(A,S)$ for honest $A$ are authenticated. The classical injective agreement is modeled by the steps highlighted in blue: when an honest agent generates a fresh key for server, it inserts it into a special set $\mathsf{begin}_i$, and whenever a server accepts a key that appears to come from an honest agent $A$, then it inserts it into a special set $\mathsf{end}$. (Note that these sets exist only in our model to specify the goals.) It is a violation of (non-injective) agreement if the server accepts a key that is not in $\mathsf{begin}_i$ (rule $R_1^2$ and rule $R_2^2$, shown here on the left respectively on the right):

```
$\forall A \in \mathsf{Hon}, S \in \mathsf{Ser}.$
    $1\colon \mathsf{receive}(\mathsf{sign}(\mathsf{inv}(PK),$
                        $\mathsf{pair}(A, NPK))).$
    $\star\colon PK \,\dot{\in}\, \mathsf{valid}(A, S).$
    $\star\colon \forall A', S'.\ NPK \,\dot{\notin}\, \mathsf{valid}(A', S').$
    $1\colon \forall A', S'.\ NPK \,\dot{\notin}\, \mathsf{revoked}(A', S').$
    $\star\colon NPK \,\dot{\notin}\, \mathsf{begin}_1(A, S).$
    $1\colon \mathsf{send}(\mathsf{attack}_1)$
```

```
$\forall A \in \mathsf{Hon}, S \in \mathsf{Ser}.$
    $2\colon \mathsf{receive}(\mathsf{crypt}(PK,$
                  $\mathsf{update}(A, S, NPK,$
                        $\mathsf{pw}(A, S)))).$
    $2\colon PK \,\dot{\in}\, \mathsf{pubkeys}(S).$
    $2\colon \forall S'.\ NPK \,\dot{\notin}\, \mathsf{pubkeys}(S').$
    $2\colon \forall A', S'.\ NPK \,\dot{\notin}\, \mathsf{seen}(A', S').$
    $\star\colon NPK \,\dot{\notin}\, \mathsf{begin}_2(A, S).$
    $2\colon \mathsf{send}(\mathsf{attack}_2)$
```

and of injective agreement (i.e. replay) if the server accepts a key that is already in $\mathsf{end}_i$ (rule $R_1^3$ to the left and rule $R_2^3$ to the right):

```
$\forall A \in \mathsf{Hon}, S \in \mathsf{Ser}.$
    $1\colon \mathsf{receive}(\mathsf{sign}(\mathsf{inv}(PK),$
                        $\mathsf{pair}(A, NPK))).$
    $\star\colon PK \,\dot{\in}\, \mathsf{valid}(A, S).$
    $\star\colon \forall A', S'.\ NPK \,\dot{\notin}\, \mathsf{valid}(A', S').$
    $1\colon \forall A', S'.\ NPK \,\dot{\notin}\, \mathsf{revoked}(A', S').$
    $\star\colon NPK \,\dot{\in}\, \mathsf{begin}_1(A, S).$
    $\star\colon NPK \,\dot{\in}\, \mathsf{end}_1(A, S).$
    $1\colon \mathsf{send}(\mathsf{attack}_1)$
```

```
$\forall A \in \mathsf{Hon}, S \in \mathsf{Ser}.$
    $2\colon \mathsf{receive}(\mathsf{crypt}(PK,$
                  $\mathsf{update}(A, S, NPK,$
                        $\mathsf{pw}(A, S)))).$
    $2\colon PK \,\dot{\in}\, \mathsf{pubkeys}(S).$
    $2\colon \forall S'.\ NPK \,\dot{\notin}\, \mathsf{pubkeys}(S').$
    $2\colon \forall A', S'.\ NPK \,\dot{\notin}\, \mathsf{seen}(A', S').$
    $\star\colon NPK \,\dot{\in}\, \mathsf{begin}_2(A, S).$
    $\star\colon NPK \,\dot{\in}\, \mathsf{end}_2(A, S).$
    $2\colon \mathsf{send}(\mathsf{attack}_2)$
```

Such a specification is more declarative when one clearly separates the protocol rules from the attack rules, but that has one drawback: if the protocol indeed had an attack, then one would allow the server to actually insert an unauthenticated key into its database and then in the next step the attack rule fires. For the composition result, however, we want that each protocol can rely on the other protocols to never insert unauthenticated keys into the database. This is why we integrate in rules $R_1^6$ and $R_2^6$ the checks that we are in an authenticated case (in all other cases, the rules $R_i^2$ or $R_i^3$ fire). This is similar to a "lookahead" where we prevent the execution of a transition if it leads to an attack, and directly trigger an attack. Of course, a more high-level formalism for specifying protocols may allow for a more declarative way to specify this authentication and "compile it down" to this kind of specification.

## 5 The Compositionality Results

With stateful protocols and parallel composition defined we can now formally define the concepts underlying our results and state our compositionality theorems. We first provide a result on the level of constraints and afterwards show our main theorems for stateful protocols.

### 5.1 Protocol Abstraction

Note that all steps containing the valid set family in our keyserver example have been labeled with $\star$. Labeling operations on the shared sets with $\star$ is actually an important part of our compositionality result and we now explain why.

Essentially, compositionality results aim to prevent that attacks can arise from the composition itself, i.e., attacks that do not similarly work on the components in isolation. Thus we want to show that attacks on the composed system can be sufficiently decomposed into attacks on the components. This however cannot directly work if the components have shared sets like valid in the example: if one protocol inserts something to a set and the other protocol reads from the set, then this trace in general does not have a counter-part in the second protocol alone. We thus need a kind of *interface* to how the two protocols can influence their shared sets. In the key server example, both protocols can insert public keys into the shared set valid, the first protocol can even remove them. The idea is now that we develop from each protocol an *abstract* version that subsumes all the modifications that the concrete protocol can perform on the shared sets. This can be regarded as a "contract" for the composition: each protocol *guarantees* that it will not make any modifications that are not covered by its abstract protocol, and it will *assume* that the other protocol only makes modifications covered by the other protocol's abstraction. We will still have to verify that each individual protocol is also secure when running together with the other abstract protocol, but this is in general much simpler than the composition of the two concrete protocols. (In the special case that the protocols share no sets, i.e. like in all previous composition results, the abstractions are empty, i.e., we have to verify only the individual components.)

In general, the abstraction of a component protocol $P$ is defined by restriction to those steps that are labeled $\star$, i.e., $P^\star$. We require that at least the modification of shared sets are labeled $\star$. In the keyserver example we have also labeled the operations on the authentication-related sets with a $\star$ (everything highlighted in blue): we need to ensure that we insert into the set of valid keys of an honest agent only those keys that really have been created by that agent and that have not been previously inserted. So the contract between the two protocols is that they only insert keys that are properly authenticated, but the abstraction ignores how each protocol achieves the authentication (e.g. signatures vs. passwords and seen-set). There are also some outgoing messages labeled with $\star$ which we discuss a little below.[3]

[3] In general, we will require well-formedness of the $\star$-projected protocols; this can however easily be achieved by verification tools as a transparent service.

### 5.2 Shared Terms

Before giving the compositionality conditions we first formally define what terms can be shared: Every term $t$ that occurs in multiple component protocols must be either a *basic public term* (meaning that the intruder can derive $t$ without prior knowledge, i.e., $\emptyset \vdash t$) or a *secret*. If the intruder learns a shared secret then it is considered a violation of secrecy in *all* component protocols. For instance, agent names are usually basic public terms whereas private keys are secrets. In fact, we will have that *all* shared terms (except basic public terms) are by default secrets, even public keys, until they are declassified, as explained below.

Let $Sec$ be a set of ground terms, representing the initially shared secrets of the protocols. Note that the set of shared secrets $Sec$ is not a fixed predefined set of terms, but rather just a parameter to our compositionality condition. We require that all shared terms of the protocols are either in $Sec$ or basic public terms. To precisely define this requirement, we first define the *ground sub-message patterns (GSMP)* of a set of terms $M$ as $GSMP(M) \equiv \{t \in SMP(M) \mid fv(t) = \emptyset\}$. This definition is extended to constraints $\mathcal{A}$ as the set $GSMP(\mathcal{A}) \equiv GSMP(trms(\mathcal{A}) \cup setops(\mathcal{A}))$, and similarly for protocols. To make matters smooth, we also require that $Sec \cup \{t \mid \emptyset \vdash t\}$ is closed under subterms (which is trivially the case for the basic public terms).

*Example 2.* The set $GSMP(\mathcal{P}_{ks,1} \parallel \mathcal{P}^{\star}_{ks,2})$ consists of the following set closed under subterms:

$$\begin{aligned}&\{\mathsf{attack}_1, (pk, \mathsf{ring}(a)), (pk, \mathsf{valid}(a, s)), (pk, \mathsf{revoked}(a, s)), (pk, \mathsf{begin}_i(a, s)),\\ &\quad (pk, \mathsf{end}_i(a, s)), \mathsf{sign}(\mathsf{inv}(pk), \mathsf{pair}(a, npk)) \mid i \in \{1, 2\}, pk, npk, a, s \in \mathcal{C},\\ &\quad \Gamma(\{pk, npk\}) = \{\mathsf{PubKey}\}, \Gamma(\{a, s\}) = \{\mathsf{Agent}\}\}\end{aligned}$$

and $GSMP(\mathcal{P}^{\star}_{ks,1} \parallel \mathcal{P}_{ks,2})$ consists of the following set closed under subterms:

$$\begin{aligned}&\{\mathsf{attack}_2, (pk, \mathsf{valid}(a, s)), (pk, \mathsf{seen}(a, s)), (pk, \mathsf{begin}_i(a, s)), (pk, \mathsf{end}_i(a, s)),\\ &\quad (pk, \mathsf{pubkeys}(s)), \mathsf{inv}(pk), \mathsf{crypt}(pk, \mathsf{update}(a, s, npk, \mathsf{pw}(a, s))) \mid i \in \{1, 2\},\\ &\quad pk, npk, a, s \in \mathcal{C}, \Gamma(\{pk, npk\}) = \{\mathsf{PubKey}\}, \Gamma(\{a, s\}) = \{\mathsf{Agent}\}\}\end{aligned}$$

The terms that can be shared between protocols then lie in the intersection of the ground sub-message patterns of each protocol:

**Definition 4 (GSMP disjointedness).** *Given two sets of terms $M_1$ and $M_2$, and a ground set of terms $Sec$ (the shared secrets), we say that $M_1$ and $M_2$ are* $Sec$-GSMP disjoint *iff $GSMP(M_1) \cap GSMP(M_2) \subseteq Sec \cup \{t \mid \emptyset \vdash t\}$. This is extended to constraints and protocols as expected.*

### 5.3 Declassification and Leaking

Up until now the set of shared secrets has been static. We now remove this restriction by introducing a notion of declassification that will allow shared secrets to become public during protocol execution. For instance, in protocol $\mathcal{P}_{ks,1}$ we give revoked private keys of the form $\mathsf{inv}(PK)$ to the intruder by transmitting them over the network: $\mathsf{send}(\mathsf{inv}(PK))$. The transmitted key $\mathsf{inv}(PK)$ should no longer be secret after transmission and so we call such steps for *declassification*. Since declassification involves shared secrets we require that they occur in all component protocols at the same point in time. Thus we label them with $\star$.

For any constraint $\mathcal{A}$ with model $\mathcal{I}$ we can now formally define the set of secrets that has been declassified in $\mathcal{A}$ under $\mathcal{I}$:

**Definition 5 (Declassification).** *Let $\mathcal{A}$ be a labeled constraint and $\mathcal{I}$ a model of $\mathcal{A}$. Then declassified$(\mathcal{A}, \mathcal{I}) \equiv \mathcal{I}(\{t \mid \star\colon \mathsf{receive}(t)$ occurs in $\mathcal{A}\})$ is the set of* declassified secrets of $\mathcal{A}$ under $\mathcal{I}$.

For instance, given a protocol $\mathcal{P}$, a reachable constraint $\mathcal{A}$ (i.e., $0 \Rightarrow^{\bullet\star}_{\mathcal{P}} \mathcal{A}$), and a model $\mathcal{I}$ of $\mathcal{A}$, then $\mathcal{I}(\mathcal{A})$ represents a concrete protocol run and the set *declassified*$(\mathcal{A}, \mathcal{I})$ represents the messages that have been declassified by honest-agents during the protocol run. Note that we in this definition have reversed the direction of the declassification transmission, because the $\mathsf{send}$ and $\mathsf{receive}$ steps of reachable constraints are duals of the transaction rules they originated from.

Declassification allows us to also share terms that contain secrets but are not themselves secret. For instance, public key certificates can be shared by modeling them as shared secrets that are declassified when first transmitted.

Finally, if the intruder learns a secret that has not been declassified then it counts as an attack. We say that protocol $\mathcal{P}$ *leaks* a secret $s$ if there is a reachable satisfiable constraint $\mathcal{A}$ where the intruder learns $s$ before it is declassified:

**Definition 6 (Leakage).** *Let Sec be a set of secrets and $\mathcal{I}$ be a model of the labeled constraint $\mathcal{A}$. $\mathcal{A}$* leaks a secret from *Sec* under $\mathcal{I}$ *iff there exists $s \in$ Sec $\setminus$ declassified$(\mathcal{A}, \mathcal{I})$ such that $\mathcal{I} \models \mathcal{A}|_1.\mathsf{send}(s)$ or $\mathcal{I} \models \mathcal{A}|_2.\mathsf{send}(s)$.*

Our notion of leakage requires that one of the components in isolation leaks a secret. This is important for our compositionality result later—we will require protocols not to leak in isolation (which can be verified on the protocols in isolation) for the composition to work. Note also that the set *declassified*$(\mathcal{A}, \mathcal{I})$ is unchanged during projection of $\mathcal{A}$, and so it suffices to pick the leaked $s$ from the set *Sec* $\setminus$ *declassified*$(\mathcal{A}, \mathcal{I})$ instead of *Sec* $\setminus$ *declassified*$(\mathcal{A}|_i, \mathcal{I})$.

*Example 3.* The terms occurring in the GSMP intersection of the two keyserver protocols are (a) public keys $\mathsf{pk}$, (b) private keys of the form $\mathsf{inv}(\mathsf{pk})$, (c) agent names, and (d) operations on the shared set families $\mathsf{valid}$, $\mathsf{begin}_i$, and $\mathsf{end}_i$. Agent names are basic public terms in our example, i.e., $\emptyset \vdash \mathsf{a}$ for all constants $\mathsf{a}$ of type $\mathsf{Agent}$. The public keys are not initially available to the intruder, but secrets and we declassify them whenever they are generated. To satisfy GSMP disjointedness of $\mathit{GSMP}(\mathcal{P}_{ks,1} \parallel \mathcal{P}^{\star}_{ks,2})$ and $\mathit{GSMP}(\mathcal{P}^{\star}_{ks,1} \parallel \mathcal{P}_{ks,2})$ it thus suffices to choose the following set as the set of shared secrets (where each $\mathsf{sec}_f$ is here a special secret constant used in the encoding of the private function symbol $f$):

$$\begin{aligned} Sec = \{&pk, \mathsf{inv}(pk), (pk, f(a,s)), f(a,s), \mathsf{sec}_{\mathsf{inv}}, \mathsf{sec}_f \mid \Gamma(\{a,s\}) = \{\mathsf{Agent}\},\\ &\Gamma(pk) = \mathsf{PubKey}, f \in \{\mathsf{valid}, \mathsf{begin}_1, \mathsf{end}_1, \mathsf{begin}_2, \mathsf{end}_2\}, pk, a, s \in \mathcal{C}\} \end{aligned}$$

Note that we want the set symbols like $\mathsf{valid}$ to be private. This is because terms like $\mathsf{valid}(A, S)$ occurs in both component protocols and so we have to prevent the intruder from constructing them.

### 5.4 Parallel Compositionality for Constraints

With these concepts defined we can list the requirements on constraints that are necessary to apply our result on the constraint level:

**Definition 7 (Parallel composability).** *Let $\mathcal{A}$ be a constraint and let $Sec$ be a ground set of terms. Then $(\mathcal{A}, Sec)$ is* parallel composable *iff*

1. *$\mathcal{A}|_1$ and $\mathcal{A}|_2$ are Sec-GSMP disjoint,*
2. *for all terms $t$ the step $\star\colon \mathsf{send}(t)$ does not occur in $\mathcal{A}$,*
3. *for all $s \in Sec$ and $s' \sqsubseteq s$, either $\emptyset \vdash s'$ or $s' \in Sec$,*
4. *for all $\ell\colon (t, s), \ell'\colon (t', s') \in labeledsetops(\mathcal{A})$, if $(t, s)$ and $(t', s')$ are unifiable then $\ell = \ell'$,*
5. *$\mathcal{A}$ is type-flaw resistant and $\mathcal{A}$, $\mathcal{A}|_1$, $\mathcal{A}|_2$, and $\mathcal{A}|_\star$ are all well-formed,*

where $labeledsetops(\mathcal{A}) \equiv \{\ell\colon (t, s) \mid \ell\colon \mathsf{insert}(t, s)$ or $\ell\colon \mathsf{delete}(t, s)$ or $\ell\colon t \mathrel{\dot{\in}} s$ or $\ell\colon (\forall \bar{x}.\ t \mathrel{\dot{\notin}} s)$ occurs in $\mathcal{A}\}$.

The first requirement is the core of our compositionality result and states that the protocols can only share basic public terms and shared secrets. The second requirement ensures that $\star$ steps are only used for declassification, checks, and stateful steps. The third condition is our only requirement on the shared terms; it ensures that the set $Sec \cup \{t \mid \emptyset \vdash t\}$ is closed under subterms. The fourth condition is our requirement on stateful protocols; it implies that shared sets must be labeled with a $\star$. Finally, the last condition is needed to apply the typing result and it is orthogonal to the other conditions; it is indeed only necessary so that we can apply Theorem 1 and restrict ourselves to well-typed attacks only. Typing results with different requirements could potentially be used instead. Note that we require well-formedness of *all* projections of $\mathcal{A}$. This is because we usually consider constraints reachable in composed and augmented protocols, and we need well-formedness to apply the typing result to these constraints.

With these requirements defined we can state our main result on constraints:

**Theorem 2** *If $(\mathcal{A}, Sec)$ is parallel composable and $\mathcal{I} \models \mathcal{A}$ then there exists a well-typed interpretation $\mathcal{I}_\tau$ such that either $\mathcal{I}_\tau \models \mathcal{A}|_1$ and $\mathcal{I}_\tau \models \mathcal{A}|_2$ or some prefix $\mathcal{A}'$ of $\mathcal{A}$ leaks a secret from Sec under $\mathcal{I}_\tau$.*

That is, we can obtain a well-typed model of projections $\mathcal{A}|_1$ and $\mathcal{A}|_2$ for satisfiable parallel composable constraints $\mathcal{A}$—or one of the projections has leaked a secret. In other words, if we can verify that a parallel composable constraint $\mathcal{A}$ does not have any well-typed model of both projections, and no prefix of $\mathcal{A}$ leaks a secret under any well-typed model, then it is unsatisfiable.

### 5.5 Parallel Compositionality for Protocols

Until now our parallel compositionality result has been stated on the level of constraints. As a final step we now explain how we can use Theorem 2 to prove a parallel compositionality result for protocols.

First, we define the *traces* of $\mathcal{P}$ as the set of reachable constraints: $\mathit{traces}(\mathcal{P}) \equiv \{\mathcal{A} \mid 0 \Rightarrow^{\bullet\star}_{\mathcal{P}} \mathcal{A}\}$. Then $(\mathcal{P}_1 \parallel \mathcal{P}_2, \mathit{Sec})$ is *parallel composable* if $\mathcal{P}_1 \parallel \mathcal{P}_2$ is type-flaw resistant and $(\mathcal{A}, \mathit{Sec})$ is parallel composable for all $\mathcal{A} \in \mathit{traces}(\mathcal{P}_1 \parallel \mathcal{P}_2)$. For protocols we need to require that their composition is type-flaw resistant. It is not sufficient to simply require it for the component protocols in isolation; unifiable messages from different protocols might break type-flaw resistance otherwise.

*Example 4.* Continuing example 3 we now show that all reachable constraints of $\mathcal{P}_{ks,1} \parallel \mathcal{P}_{ks,2}$ satisfy conditions one to four of Definition 7. A similar keyserver protocol [16] has previously been shown to be type-flaw resistant and well-formed and so we focus on the remaining conditions here. Any constraint $\mathcal{A} \in \mathit{traces}(\mathcal{P}_{ks,1} \parallel \mathcal{P}_{ks,2})$ consists of a sequence of (the duals of) transaction strands from $\mathcal{P}_{ks,1} \parallel \mathcal{P}_{ks,2}$. GSMP disjointedness of $\mathcal{A}$ therefore follows from GSMP disjointedness of the keyserver protocols, i.e., the previous example, since $\mathcal{A}|_1 \in \mathit{traces}(\mathcal{P}_{ks,1} \parallel \mathcal{P}^{\star}_{ks,2})$ and $\mathcal{A}|_2 \in \mathit{traces}(\mathcal{P}^{\star}_{ks,1} \parallel \mathcal{P}_{ks,2})$. Hence the first condition is satisfied. Conditions two and three are satisfied since $\mathcal{P}_{ks,1} \parallel \mathcal{P}_{ks,2}$ does not contain any steps of the form $\ell\colon \mathsf{send}(t)$ and since any subterm of a term from *Sec* (as defined in the previous example) is either in *Sec* or an agent name (a basic public term). Note that $\mathit{labeledsetops}(\mathcal{A})$ consists of instances of labeled terms from the following set: $\{1\colon (PK_0, \mathsf{ring}(A_0)),\ 1\colon (PK_1, \mathsf{revoked}(A_1, S_1)),$ $2\colon (PK_2, \mathsf{seen}(A_2, S_2)),\ \star\colon (PK_3, \mathsf{valid}(A_3, S_3)),\ \star\colon (PK^i_4, \mathsf{begin}_i(A^i_4, S^i_4)),$ $\star\colon (PK^i_5, \mathsf{end}_i(A^i_5, S^i_5)) \mid i \in \{1, 2\}\}$. For all pairs $\ell\colon (t, s)$, $\ell'\colon (t', s')$ in this set we have that $\ell = \ell'$ if $(t, s)$ and $(t', s')$ are unifiable. Hence condition 4 is satisfied.

As a consequence of Theorem 2 we have that any protocol $\mathcal{P}_1$ can be safely composed with another protocol $\mathcal{P}_2$ provided that $\mathcal{P}_1 \parallel \mathcal{P}^{\star}_2$ is secure and that $\mathcal{P}^{\star}_1 \parallel \mathcal{P}_2$ does not leak a secret:

**Theorem 3** *If $(\mathcal{P}_1 \parallel \mathcal{P}_2, \mathit{Sec})$ is parallel composable, $\mathcal{P}_1 \parallel \mathcal{P}^{\star}_2$ is well-typed secure in isolation, and $\mathcal{P}^{\star}_1 \parallel \mathcal{P}_2$ does not leak a secret under any well-typed model, then all goals of $\mathcal{P}_1$ hold in $\mathcal{P}_1 \parallel \mathcal{P}_2$ (even in the untyped model).*

Note that the only requirement on protocol $\mathcal{P}_2$ is that it does not leak any secrets—it may in fact have other flaws that does not involve leakage of shared secrets and the goals of protocol $\mathcal{P}_1$ would still hold in the composition $\mathcal{P}_1 \parallel \mathcal{P}_2$. Thus, the composition of parallel composable and secure protocols is secure:

**Corollary 1.** *If $(\mathcal{P}_1 \parallel \mathcal{P}_2, \mathit{Sec})$ is parallel composable and $\mathcal{P}_1 \parallel \mathcal{P}^{\star}_2$ and $\mathcal{P}^{\star}_1 \parallel \mathcal{P}_2$ are both secure in isolation then the composition $\mathcal{P}_1 \parallel \mathcal{P}_2$ is also secure (even in the untyped model).*

## 6 Conclusion and Related Work

Our composition theorem for parallel composition is the newest in a sequence of parallel composition results that are each pushing the boundaries of the class of protocols that can be composed [14,13,2,12,8,9,7,3,1]. The first results simply require completely disjoint encryptions; subsequent results allowed the sharing of long-term keys, provided that wherever the common keys are used, the content messages of the different protocols are distinguished, for instance by tagging. Other aspects are which primitives are supported as well as what forms of negative conditions, e.g. to support as goals the full geometric fragment.

Our result lifts the common requirement that the component protocols only share a fixed set of long-term public and private constants. Our result allows for stateful protocols that maintain databases (such as a key server) and the databases may even be shared between these protocols. This includes the possibility to declassify long-term secrets, e.g., to verify that a protocol is even secure if the intruder learns all old private keys. Both databases, shared databases, and declassification are considerable generalizations over the existing results.

Like [1] our result links the parallel compositionality result with a typing result such as the result of [16], i.e., essentially requiring that all messages of different meaning have a distinguishable form. Under this requirement it is sound to restrict the intruder model to using only well-typed messages which greatly simplifies many related problems. While one may argue that such a typing result is not strictly necessary for composition, we believe it is good practice and also fits well with disjointness requirements of parallel composition. Moreover, many existing protocols already satisfy our typing requirement, since, unlike tagging schemes, this does not require a modification of a protocol as long as there is some way to distinguish messages of different meaning.

There are other types of compositionality results for sequential and vertical composition, where the protocols under composition do build upon each other, e.g., one protocol establishes a key that is then subsequently used by another protocol [2,10,9,6,18,11]. This requires that one protocol satisfies certain properties (e.g. that the key exchange is authenticated and secret) for the other protocol to rely on. Our composition result allows for such sequential composition through shared databases: a key exchange protocol may enter keys into a shared set, and the other protocol consumes these keys. Thus our concept of sharing sets generalizes the interactions between otherwise independent protocols, and one only needs to think about the interface (e.g., only authenticated, fresh, secret keys can be entered into the database; they can only be used once). Moreover, we believe that sets are also a nice way to talk about this interaction.

There are several interesting aspects of compositionality that our result does not cover, for instance, [7] discusses the requirements for composing password-based protocols, and [3] investigates conditions under which privacy properties can be preserved under protocol composition.

So far, compositionality results are solely "paper-and-pencil" proofs. The proof arguments are often quite subtle, e.g., given an attack where the intruder learned a nonce from one protocol and uses it in another protocol, one has to prove that the attack does not rely on this, but would similarly work for distinct nonces. It is not uncommon that parts of such proofs are a bit sketchy with the danger of overlooking some subtle problems as for instance described in [15]. For this reason, we have formalized the compositionality result—on the level of ordinary constraints—in the proof assistant Isabelle/HOL [19], extending the formalization of [15], giving the extremely high correctness guarantee of machine-checked proofs. To our knowledge, this work is the first such formalization of a compositionality result in a proof assistant, with the notable exception of a study in Isabelle/HOL of compositional reasoning on concrete protocols [5].

Finally, all the works discussed so far are based on a black-box model of cryptography. There are several cryptographic frameworks for composition, most notably universal composability, reactive simulatability [4], and [17]. Considering the real cryptography makes compositional reasoning several orders of magnitude harder than abstract cryptography models. It is an intriguing question whether stateful protocol composition can be lifted to the full cryptographic level.

## References

1. O. Almousa, S. Mödersheim, P. Modesti, and L. Viganò. Typing and compositionality for security protocols: A generalization to the geometric fragment. In *ESORICS 2015*, pages 209–229, 2015.
2. S. Andova, C. J. F. Cremers, K. Gjøsteen, S. Mauw, S. F. Mjølsnes, and S. Radomirović. A framework for compositional verification of security protocols. *Inf. Comput.*, 206(2-4):425–459, 2008.
3. M. Arapinis, V. Cheval, and S. Delaune. Composing security protocols: From confidentiality to privacy. In R. Focardi and A. Myers, editors, *Principles of Security and Trust*, pages 324–343, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
4. M. Backes, B. Pfitzmann, and M. Waidner. The reactive simulatability (RSIM) framework for asynchronous systems. *Inf. Comput.*, 205(12):1685–1720, 2007.
5. D. F. Butin. *Inductive analysis of security protocols in Isabelle/HOL with applications to electronic voting.* PhD thesis, Dublin City University, Nov. 2012.
6. V. Cheval, V. Cortier, and B. Warinschi. Secure composition of pkis with public key protocols. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 144–158, Aug 2017.
7. C. Chevalier, S. Delaune, S. Kremer, and M. D. Ryan. Composition of password-based protocols. *Formal Methods in System Design*, 43(3):369–413, Dec 2013.
8. V. Cortier and S. Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1):1–36, 2009.
9. Ştefan Ciobâcă and V. Cortier. Protocol composition for arbitrary primitives. In *CSF*, pages 322–336. IEEE, 2010.
10. S. Escobar, C. A. Meadows, J. Meseguer, and S. Santiago. Sequential protocol composition in maude-npa. In *Computer Security - ESORICS 2010, 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings*, pages 303–318, 2010.
11. T. Groß and S. Mödersheim. Vertical protocol composition. In *Computer Security Foundations Symposium (CSF), 2011 IEEE 24th*, pages 235 –250, june 2011.
12. J. D. Guttman. Cryptographic Protocol Composition via the Authentication Tests. In *FOSSACS'09*, pages 303–317. Springer, 2009.

13. J. D. Guttman and F. J. Thayer. Protocol independence through disjoint encryption. In *CSFW*, pages 24–34. IEEE, 2000.
14. N. Heintze and J. D. Tygart. A model for secure protocols and their compositions. In *Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 2–13, May 1994.
15. A. V. Hess and S. Mödersheim. Formalizing and Proving a Typing Result for Security Protocols in Isabelle/HOL. In *CSF 2017*, 2017.
16. A. V. Hess and S. Mödersheim. A Typing Result for Stateful Protocols. In *CSF 2018*, 2018. To appear, preprint available at `https://people.compute.dtu.dk/samo/SPC.pdf`.
17. R. Küsters and M. Tuengerthal. Composition theorems without pre-established session identifiers. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 41–50, New York, NY, USA, 2011. ACM.
18. S. Mödersheim and L. Viganò. Secure pseudonymous channels. *ESORICS 2009*, pages 337–354, 2009.
19. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.

# A High-Level Explanation of the Proofs

In this appendix we will explain the main ideas for the proofs of our compositionality results. We have formalized in Isabelle/HOL the majority of our work, namely the compositionality result on the ordinary constraints (i.e., constraints built using only the cases $\mathsf{send}(t)$, $\mathsf{receive}(t)$, $t \doteq t'$, and $\forall \bar{x}.\ t \not\doteq t'$) and the reachable constraints of ordinary protocols. The stateful typing result is formalized in Isabelle/HOL as well. The extensions to stateful constraints and protocols have not been formalized in Isabelle/HOL yet, but the paper proofs reuses the core proof idea of the Isabelle-formalized stateful typing result to lift the compositionality results from ordinary to stateful constraints and protocols. The proofs for the theorems on the protocol level are mostly applications of the constraint level theorems and so we will not focus on those theorems in this appendix.

## A.1 Proving Theorem 2 for Ordinary Constraints

For Theorem 2 we need to show that for satisfiable parallel composable constraints $\mathcal{A}$ with shared secrets $Sec$ we can obtain a well-typed model of projections $\mathcal{A}|_1$ and $\mathcal{A}|_2$ or $\mathcal{A}$ has leaked a secret in one of the projections. In a nutshell we show that any term $t$ occurring in a $\ell\colon \mathsf{send}(t)$ step of $\mathcal{A}$ need only to be constructed from terms of protocol $\ell$, unless leakage has occurred previously. For that we first need a notion of terms belonging to a specific protocol:

**Definition 8.** *Let $\mathcal{A}$ be a constraint and Sec be a set of shared secrets. A term $t$ is* $i$-specific *iff $t \in GSMP(\mathcal{A}|_i) \setminus Sec \cup \{t \mid \emptyset \vdash t\}$ for a label $i$. A term $t$ is* homogeneous (w.r.t. $\mathcal{A}$ and *Sec*) *iff there exists subterms $t_1$ and $t_2$ of $t$ such that each $t_i$ for $i \in \{1, 2\}$ is $i$-specific w.r.t. $\mathcal{A}$ and Sec.*

Then all ground sub-message patterns are homogeneous:

**Lemma 1** *If $(\mathcal{A}, Sec)$ is parallel composable and $t \in GSMP(\mathcal{A})$ then $t$ is homogeneous.*

Given a constraint $\mathcal{A}$ and a set of shared secrets $Sec$ we now define a useful variant $\vdash_{\text{hom}}^{\mathcal{A},Sec}$ of the intruder deduction relation $\vdash$ as the restriction of $\vdash$ to homogeneous terms only. This relation satisfies a useful property:

**Lemma 2** *Let $(\mathcal{A}, Sec)$ be parallel composable and $t$ be a homogeneous term. Then $ik(\mathcal{A}) \vdash t$ iff $ik(\mathcal{A}) \vdash_{\text{hom}}^{\mathcal{A},Sec} t$.*

This is useful because we can prove that all homogeneous GSMP terms can be derived purely through derivation of other homogeneous GSMP terms. In other words, for homogeneous terms such as those in parallel composable constraints we can reduce the intruder derivation problem to $\vdash_{\text{hom}}^{\mathcal{A},Sec}$.

**Lemma 3** *Let $(\mathcal{A}, Sec)$ be parallel composable, $\mathcal{I}$ be a well-typed model of $\mathcal{A}$. If $ik(\mathcal{I}(\mathcal{A})) \vdash_{\text{hom}}^{\mathcal{A},Sec} t$, then either*

- *$t \notin Sec \setminus declassified(\mathcal{A}, \mathcal{I})$, and*
- *if $i \in \{1, 2\}$ and $t \in GSMP(\mathcal{A}|_i)$ then $ik(\mathcal{A}|_i) \vdash_{\text{hom}}^{\mathcal{A},Sec} t$,*

*or there exists $s \in Sec \setminus declassified(\mathcal{A}, \mathcal{I})$ and $j \in \{1, 2\}$ s.t. $ik(\mathcal{I}(\mathcal{A}|_j)) \vdash_{\text{hom}}^{\mathcal{A},Sec} s$.*

The idea is that deriving a term $f(t_1, \ldots, t_n)$ that "falls outside of" the homogeneous GSMP terms is only possible by composition; if all the immediate subterms $t_i$ are homogeneous GSMP terms then deriving $f(t_1, \ldots, t_n)$ must have happened by an application of the (*Compose*) rule. Usually, such proofs proceed by inspecting the derivation tree of the derivation of $f(t_1, \ldots, t_n)$, and, in the case where $f(t_1, \ldots, t_n)$ has been derived from decomposition, either transforming the tree to remove unnecessary decomposition steps or regress to the first decomposition step. Such arguments are cumbersome to formalize in Isabelle/HOL since one would need a deep embedding of the derivation tree. For our purposes, however, it is sufficient to only encode the *height* of the derivation tree and so we equip the relation $\vdash$ with such a number: $M \vdash_k t$ iff $k$ is the maximum number of applications of the (*Compose*) and (*Decompose*) rules in any path of the derivation tree for $M \vdash t$. Essentially, we prove that no matter how many steps occur in the derivation tree of $f(t_1, \ldots, t_n)$ the first time the term is derived (it might have been derived later on through decomposition) is always a composition step.

With Lemma 2 and 1 we can prove a useful consequence of Lemma 3:

**Lemma 4** *Let $(\mathcal{A}, Sec)$ be parallel composable, $\mathcal{I}$ be a well-typed model of $\mathcal{A}$, $i \in \{1, 2\}$ be a label, and $t$ a term such that $t \in GSMP(\mathcal{A}|_i)$. If $ik(\mathcal{I}(\mathcal{A})) \vdash t$ then either $ik(\mathcal{I}(\mathcal{A}|_i)) \vdash t$ or $\mathcal{A}$ leaks a secret from $Sec$.*

Now we can use Lemma 4 to show that the models $\mathcal{I}$ of parallel composable constraints $\mathcal{A}$ are also models of the projections $\mathcal{A}|_i$, or some secret is leaked. The proof is by structural induction on the constraint $\mathcal{A}$. The only non-trivial case is where a step of the form $\mathsf{send}(t)$ occurs in $\mathcal{A}$, i.e., when a prefix of the form $\mathcal{A}'.\mathsf{send}(t)$ exists for $\mathcal{A}$. By the constraint semantics such a prefix corresponds to a derivation constraint $ik(\mathcal{I}(\mathcal{A}')) \vdash \mathcal{I}(t)$, and here we can apply Lemma 4. Thus:

**Lemma 5** *Let $(\mathcal{A}, Sec)$ be parallel composable and let $\mathcal{I}$ be a well-typed model of $\mathcal{A}$. Then either $\mathcal{I} \models \mathcal{A}|_1$ and $\mathcal{I} \models \mathcal{A}|_2$ or some prefix $\mathcal{A}'$ of $\mathcal{A}$ leaks a secret from Sec under $\mathcal{I}$.*

Finally, we can use Theorem 1 and Lemma 5 to relax the well-typedness assumption and prove our main result on the level of stateless constraints:

**Lemma 6** *Let $(\mathcal{A}, Sec)$ be parallel composable and let $\mathcal{I}$ be a model of $\mathcal{A}$. Then there exists a well-typed interpretation $\mathcal{I}_\tau$ of $\mathcal{A}$ such that either $\mathcal{I}_\tau \models \mathcal{A}|_1$ and $\mathcal{I}_\tau \models \mathcal{A}|_2$ or some prefix $\mathcal{A}'$ of $\mathcal{A}$ leaks a secret from Sec under $\mathcal{I}_\tau$.*

### A.2 Proving Theorem 2 for Stateful Constraints

For stateful constraints the proof idea is to use a variant of a reduction technique introduced in [16] to reduce the compositionality problem for stateful constraints to the Isabelle-formalized compositionality problem for ordinary constraints. We first make some definitions:

**Definition 9 (Projections).** *Given a finite set $D = \{\ell_1 \colon (t_1, s_1), \ldots, \ell_n \colon (t_n, s_n)\}$, where each $t_i$ and $s_i$, are terms and $\ell_i \in \{1, 2, \star\}$ are labels, we define the* projection of $D$ to $\ell$*, written $|D|_\ell$, as follows: $|D|_\ell = \{\ell' \colon d \in D \mid \ell = \ell'\}$.*

The constraint reduction $tr$ is now defined as follows:

**Definition 10 (Translation of symbolic constraints).** *Given a constraint $\mathcal{A}$ its translation into ordinary constraints is denoted by $tr(\mathcal{A}) = tr_\emptyset(\mathcal{A})$ where:*

$$tr_D(0) = \{0\}$$
$$tr_D(\ell \colon \mathsf{insert}(t, s).\mathcal{A}) = tr_{D \cup \{\ell \colon (t,s)\}}(\mathcal{A})$$
$$tr_D(\ell \colon \mathsf{delete}(t, s).\mathcal{A}) = \{$$
$$\ell \colon (t, s) \doteq d_1. \cdots .\ell \colon (t, s) \doteq d_i.\ell \colon (t, s) \not\doteq d_{i+1}. \cdots .\ell \colon (t, s) \not\doteq d_n.\mathcal{A}' \mid$$
$$|D|_\ell = \{\ell \colon d_1, \ldots, \ell \colon d_i, \ldots, \ell \colon d_n\}, 0 \leq i \leq n, \mathcal{A}' \in tr_{D \setminus \{\ell \colon d_1, \ldots, \ell \colon d_i\}}(\mathcal{A})\}$$
$$tr_D(\ell \colon t \mathrel{\dot\in} s.\mathcal{A}) = \{\ell \colon (t, s) \doteq d.\mathcal{A}' \mid \ell \colon d \in |D|_\ell, \mathcal{A}' \in tr_D(\mathcal{A})\}$$
$$tr_D(\ell \colon (\forall \bar{x}.\ t \mathrel{\dot\notin} s).\mathcal{A}) = \{\ell \colon (\forall \bar{x}.\ (t, s) \not\doteq d_1). \cdots .\ell \colon (\forall \bar{x}.\ (t, s) \not\doteq d_n).\mathcal{A}' \mid$$
$$|D|_\ell = \{\ell \colon d_1, \ldots, \ell \colon d_n\}, 0 \leq n, \mathcal{A}' \in tr_D(\mathcal{A})\}$$
$$tr_D(\ell \colon \mathfrak{a}.\mathcal{A}) = \{\ell \colon \mathfrak{a}.\mathcal{A}' \mid \mathcal{A}' \in tr_D(\mathcal{A})\} \textit{ otherwise}$$

Note that we apply projections $|D|_\ell$ when translation set operations with label $\ell$. Hence we never have "mix" two set operations with different labels in the reduction. A crucial point here is that parallel compositionality makes such mixing unnecessary, and this enables us to prove a strong relationship between translated constraints and projections:

**Lemma 7** *Let $i \in \{1, 2\}$ be a label. If $\mathcal{B} \in tr_D(\mathcal{A})$ then $\mathcal{B}|_i \in tr_{|D|_i \cup |D|_\star}(\mathcal{A}|_i)$.*

Now the core idea is to reduce the compositionality problem on stateful constraints to ordinary constraints using the translation $tr$. For that reason we need to show that the translation is correct, i.e., that the set of models of the input constraint is exactly the set of models of the translation:

**Lemma 8 (Semantic equivalence of reduction)** *Let $\mathcal{A}$ be a constraint and $D = \{\ell_1 \colon (t_1, s_1), \ldots, \ell_n \colon (t_n, s_n)\}$. Assume that all unifiable set operations occurring in $\mathcal{A}$ and $D$ carry the same label, i.e., if $\ell \colon (t, s), \ell' \colon (t', s') \in \mathit{labeledsetops}(\mathcal{A}) \cup D$ and $\exists \delta.\ \delta((t, s)) = \delta((t', s'))$ then $\ell = \ell'$. Assume also that the set of variables occurring in $D$ is disjoint from the bound variables of $\mathcal{A}$. Then the models of $\mathcal{A}$ are the same as the models of $\mathit{tr}(\mathcal{A})$, i.e., $[\![M, \mathcal{I}(D); \mathcal{A}]\!]\ \mathcal{I}$ iff there exists $\mathcal{B} \in \mathit{tr}_D(\mathcal{A})$ such that $[\![M, \emptyset; \mathcal{B}]\!]\ \mathcal{I}$.*

This statement is very similar to Theorem 2 of [16] and can be proven using it or similarly. Note that the first assumption of Lemma 10 is similar to Definition 7(4).

For proving Theorem 2 we now only need to lift Lemma 6 to stateful constraints. That is, given $\mathcal{I} \models \mathcal{A}$ we obtain $\mathcal{B} \in \mathit{tr}(\mathcal{A})$ such that $\mathcal{I} \models \mathcal{B}$. For $\mathcal{B}$ we can apply Lemma 6; either $\mathcal{I}_\tau \models \mathcal{B}|_i$ for all $i \in \{1, 2\}$ or $\mathcal{B}$ leaks, for some well-typed interpretation $\mathcal{I}_\tau$. Finally, with Lemma 10 and 11 we can show that either $\mathcal{I}_\tau \models \mathcal{A}|_i$ for all $i \in \{1, 2\}$ or $\mathcal{A}$ leaks. Thus:

**Theorem 2.** *If $(\mathcal{A}, \mathit{Sec})$ is parallel composable and $\mathcal{I} \models \mathcal{A}$ then there exists a well-typed interpretation $\mathcal{I}_\tau$ such that either $\mathcal{I}_\tau \models \mathcal{A}|_1$ and $\mathcal{I}_\tau \models \mathcal{A}|_2$ or some prefix $\mathcal{A}'$ of $\mathcal{A}$ leaks a secret from $\mathit{Sec}$ under $\mathcal{I}_\tau$.*

## B Proofs of Theorem 2 and Theorem 3 for Stateful Constraints and Stateful Protocols

We have proven, in Isabelle/HOL, Theorem 2 and Theorem 3 for ordinary constraints (i.e., constraints built using only the cases $\mathsf{send}(t)$, $\mathsf{receive}(t)$, $t \doteq t'$, and $\forall \bar{x}.\ t \not\doteq t'$). In this appendix we will show how to lift these theorems to stateful constraints (and stateful protocols) using a variant of a reduction technique introduced in [16].

### B.1 Definitions

We first make some definitions and abbreviations:

**Definition 11 (Projections).** *Given a finite set $D = \{\ell_1 \colon (t_1, s_1), \ldots, \ell_n \colon (t_n, s_n)\}$, where each $t_i$ and $s_i$, are terms and $\ell_i \in \{1, 2, \star\}$ are labels, we define the* projection of $D$ to $\ell$, *written $|D|_\ell$, as follows: $|D|_\ell = \{\ell' \colon d \in D \mid \ell = \ell'\}$.*

**Definition 12 (Translation of symbolic constraints).** *Given a constraint $\mathcal{A}$ its translation into ordinary constraints is denoted by $tr(\mathcal{A}) = tr_{\emptyset}(\mathcal{A})$ where:*

$$\begin{array}{l}
tr_D(0) = \{0\} \\
tr_D(\ell\colon \mathsf{insert}(t,s).\mathcal{A}) = tr_{D \cup \{\ell\colon (t,s)\}}(\mathcal{A}) \\
tr_D(\ell\colon \mathsf{delete}(t,s).\mathcal{A}) = \{ \\
\quad \ell\colon (t,s) \doteq d_1.\cdots.\ell\colon (t,s) \doteq d_i.\ell\colon (t,s) \not\doteq d_{i+1}.\cdots.\ell\colon (t,s) \not\doteq d_n.\mathcal{A}' \mid \\
\quad |D|_\ell = \{\ell\colon d_1, \ldots, \ell\colon d_i, \ldots, \ell\colon d_n\}, 0 \leq i \leq n, \mathcal{A}' \in tr_{D \setminus \{\ell\colon d_1, \ldots, \ell\colon d_i\}}(\mathcal{A})\} \\
tr_D(\ell\colon t \mathrel{\dot{\in}} s.\mathcal{A}) = \{\ell\colon (t,s) \doteq d.\mathcal{A}' \mid \ell\colon d \in |D|_\ell, \mathcal{A}' \in tr_D(\mathcal{A})\} \\
tr_D(\ell\colon (\forall \bar{x}.\ t \mathrel{\dot{\notin}} s).\mathcal{A}) = \{\ell\colon (\forall \bar{x}.\ (t,s) \not\doteq d_1).\cdots.\ell\colon (\forall \bar{x}.\ (t,s) \not\doteq d_n).\mathcal{A}' \mid \\
\quad |D|_\ell = \{\ell\colon d_1, \ldots, \ell\colon d_n\}, 0 \leq n, \mathcal{A}' \in tr_D(\mathcal{A})\} \\
tr_D(\ell\colon \mathfrak{a}.\mathcal{A}) = \{\ell\colon \mathfrak{a}.\mathcal{A}' \mid \mathcal{A}' \in tr_D(\mathcal{A})\} \ \textit{otherwise}
\end{array}$$

We will also use the following abbreviations for arbitrary protocols $\mathcal{P}_1$, $\mathcal{P}_2$:

1. $\mathcal{P}_1^\bullet \equiv \mathcal{P}_1 \parallel \mathcal{P}_2^\star$
2. $\mathcal{P}_2^\bullet \equiv \mathcal{P}_1^\star \parallel \mathcal{P}_2$
3. $\mathfrak{P}^\bullet \equiv \{\mathcal{A} \mid \mathcal{A}|_1 \in traces(\mathcal{P}_1^\bullet), \mathcal{A}|_2 \in traces(\mathcal{P}_2^\bullet)\}$

### B.2 Proof Sketches

For the constraint level the core idea is to reduce the compositionality problem on stateful constraints to ordinary constraints using the translation $tr$. For that reason we need to show that the translation is correct (Lemma 10), i.e., that the set of models of the input constraint is exactly the set of models of the translation.

By a straightforward induction proof over the structure of constraints we can prove that $tr$ preserves the properties we need for our compositionality result:

**Lemma 9 (Preservation of well-formedness and parallel compositionality)** *If $\mathcal{A}$ is well-formed and parallel composable, and if $\mathcal{B} \in tr(\mathcal{A})$, then $\mathcal{B}$ is well-formed and parallel composable.*

**Lemma 10 (Semantic equivalence of reduction)** *Let $\mathcal{A}$ be a constraint and $D = \{\ell_1\colon (t_1,s_1), \ldots, \ell_n\colon (t_n,s_n)\}$. Assume that all unifiable set operations occurring in $\mathcal{A}$ and $D$ carry the same label, i.e., if $\ell\colon (t,s), \ell'\colon (t',s') \in labeledsetops(\mathcal{A}) \cup D$. and $\exists \delta.\ \delta((t,s)) = \delta((t',s'))$ then $\ell = \ell'$. Assume also that the set of variables occurring in $D$ is disjoint from the bound variables of $\mathcal{A}$. Then the models of $\mathcal{A}$ are the same as the models of $tr(\mathcal{A})$, i.e., $[\![M, \mathcal{I}(D); \mathcal{A}]\!]\ \mathcal{I}$ iff there exists $\mathcal{B} \in tr_D(\mathcal{A})$ such that $[\![M, \emptyset; \mathcal{B}]\!]\ \mathcal{I}$.*

*Proof.* For this proof let us first define the following variant of $tr$ where we in the $\mathsf{delete}$, $\dot{\in}$, and $\dot{\notin}$ cases do not project $D$ to the current label $\ell$ (in contrast to

$tr$):

$tr'_D(0) = \{0\}$
$tr'_D(\ell\colon \mathsf{insert}(t,s).\mathcal{A}) = tr'_{D\cup\{\ell\colon (t,s)\}}(\mathcal{A})$
$tr'_D(\ell\colon \mathsf{delete}(t,s).\mathcal{A}) = \{$
$\quad \ell_1\colon (t,s) \doteq d_1.\cdots.\ell_i\colon (t,s) \doteq d_i.\ell_{i+1}\colon (t,s) \dot{\neq} d_{i+1}.\cdots.\ell_n\colon (t,s) \dot{\neq} d_n.\mathcal{A}' \mid$
$\quad D = \{\ell_1\colon d_1, \ldots, \ell_i\colon d_i, \ldots, \ell_n\colon d_n\}, 0 \leq i \leq n, \mathcal{A}' \in tr'_{D\setminus\{\ell_1\colon d_1,\ldots,\ell_i\colon d_i\}}(\mathcal{A})\}$
$tr'_D(\ell\colon t \mathrel{\dot{\in}} s.\mathcal{A}) = \{\ell'\colon (t,s) \doteq d.\mathcal{A}' \mid \ell'\colon d \in D, \mathcal{A}' \in tr'_D(\mathcal{A})\}$
$tr'_D(\ell\colon (\forall\bar{x}.\ t \mathrel{\dot{\notin}} s).\mathcal{A}) = \{\ell_1\colon (\forall\bar{x}.\ (t,s) \dot{\neq} d_1).\cdots.\ell_n\colon (\forall\bar{x}.\ (t,s) \dot{\neq} d_n).\mathcal{A}' \mid$
$\quad D = \{\ell_1\colon d_1, \ldots, \ell_n\colon d_n\}, 0 \leq n, \mathcal{A}' \in tr'_D(\mathcal{A})\}$
$tr'_D(\ell\colon \mathfrak{a}.\mathcal{A}) = \{\ell\colon \mathfrak{a}.\mathcal{A}' \mid \mathcal{A}' \in tr'_D(\mathcal{A})\}$ otherwise

The theorem follows from the following two statements (the assumptions of this lemma still apply to $D$ and $\mathcal{A}$):

$$\llbracket M, \mathcal{I}(D); \mathcal{A}\rrbracket\ \mathcal{I} \text{ iff } (\exists \mathcal{B}' \in tr'_D(\mathcal{A}).\ \llbracket M, \emptyset; \mathcal{B}'\rrbracket\ \mathcal{I}) \tag{1}$$

$$(\exists \mathcal{B} \in tr_D(\mathcal{A}).\ \llbracket M, \emptyset; \mathcal{B}\rrbracket\ \mathcal{I}) \text{ iff } (\exists \mathcal{B}' \in tr'_D(\mathcal{A}).\ \llbracket M, \emptyset; \mathcal{B}'\rrbracket\ \mathcal{I}) \tag{2}$$

Statement (1) is actually a simple adaption of Theorem 2 of [16]. The rest of this proof is to show statement (2) and we prove it by proving each direction of the bi-implication. Both proofs are by induction over the structure of $\mathcal{A}$ and we give the proof only for the most difficult case: delete. All remaining cases are similar. Note that the assumptions of this lemma still apply, but we will skip proving the antecedents of any induction hypothesis we use since those proofs are trivial.

1. To show:

   *If $\mathcal{B} \in tr_D(\mathcal{A})$ and $\llbracket M, \emptyset; \mathcal{B}\rrbracket\ \mathcal{I}$ then $\llbracket M, \emptyset; \mathcal{B}'\rrbracket\ \mathcal{I}$ for some $\mathcal{B}' \in tr'_D(\mathcal{A})$.*

   Case $\mathcal{A} = (\ell\colon \mathsf{delete}(t,s)).\mathcal{A}_0$:
   In this case we know that $\mathcal{B}$ must be of the form:

   $$\mathcal{B} = \ell\colon (t,s) \doteq d_1.\cdots.\ell\colon (t,s) \doteq d_i.\ell\colon (t,s) \dot{\neq} d_{i+1}.\cdots.\ell\colon (t,s) \dot{\neq} d_n.\mathcal{B}_0$$

   for some $\mathcal{B}_0 \in tr_{D\setminus\{\ell\colon d_1,\ldots,\ell\colon d_i\}}(\mathcal{A}_0)$ where $|D|_\ell = \{\ell\colon d_1, \ldots, \ell\colon d_n\}$ and $0 \leq i \leq n$. We also know that $\llbracket M, \emptyset; \mathcal{B}\rrbracket\ \mathcal{I}$ and therefore $\llbracket M, \emptyset; \mathcal{B}_0\rrbracket\ \mathcal{I}$.
   From the induction hypothesis we can obtain $\mathcal{B}'_0 \in tr'_{D\setminus\{\ell\colon d_1,\ldots,\ell\colon d_i\}}(\mathcal{A}_0)$ such that $\llbracket M, \emptyset; \mathcal{B}'_0\rrbracket\ \mathcal{I}$. Now obtain $\ell_{k_1}, d_{k_1}, \ldots, \ell_{k_m}, d_{k_m}$ such that $D\setminus|D|_\ell = \{\ell_{k_1}\colon d_{k_1}, \ldots, \ell_{k_m}\colon d_{k_m}\}$. Hence $\ell \neq \ell_{k_j}$ for all $j \in \{1, \ldots, m\}$ (because $|D|_\ell$ contains exactly the elements of $D$ with label $\ell$) and so $\llbracket M, \emptyset; \ell_{k_1}\colon (t,s) \neq d_{k_1} \cdots .\ell_{k_m}\colon (t,s) \dot{\neq} d_{k_m}\rrbracket\ \mathcal{I}$ because of the unifiability assumption on the set operations of $\mathcal{A}$ and $D$. Let $\mathcal{B}' = \phi.\mathcal{B}'_0$ where

   $$\begin{aligned}\phi = \ell\colon (t,s) \doteq d_1.\cdots.\ell\colon (t,s) \doteq d_i.\ell\colon (t,s) \dot{\neq} d_{i+1}.\cdots.\\ \ell\colon (t,s) \dot{\neq} d_n.\ell_{k_1}\colon (t,s) \neq d_{k_1}.\cdots.\ell_{k_m}\colon (t,s) \neq d_{k_m}\end{aligned}$$

   We can then conclude that $\mathcal{B}' \in tr'_D(\mathcal{A})$ and $\llbracket M, \emptyset; \mathcal{B}'\rrbracket\ \mathcal{I}$.

2. To show:

   *If $\mathcal{B}' \in \mathit{tr}'_D(\mathcal{A})$ and $[\![M, \emptyset; \mathcal{B}']\!]\ \mathcal{I}$ then $[\![M, \emptyset; \mathcal{B}]\!]\ \mathcal{I}$ for some $\mathcal{B} \in \mathit{tr}_D(\mathcal{A})$.*

   Case $\mathcal{A} = (\ell\colon \mathsf{delete}(t, s)).\mathcal{A}_0$:
   In this case we know that $\mathcal{B}'$ must be of the form:

   $$\mathcal{B}' = \ell_1\colon (t, s) \doteq d_1. \cdots .\ell_i\colon (t, s) \doteq d_i.\ell_{i+1}\colon (t, s) \not\doteq d_{i+1}. \cdots .\ell_n\colon (t, s) \not\doteq d_n.\mathcal{B}'_0$$

   for some $\mathcal{B}'_0 \in \mathit{tr}'_{D\setminus\{\ell_1\colon d_1,\ldots,\ell_i\colon d_i\}}(\mathcal{A}_0)$ where $D = \{\ell_1\colon d_1, \ldots, \ell_n\colon d_n\}$ and $0 \leq i \leq n$. We also know that $[\![M, \emptyset; \mathcal{B}']\!]\ \mathcal{I}$ and therefore $[\![M, \emptyset; \mathcal{B}'_0]\!]\ \mathcal{I}$. Since $(t, s)$ and $d'$ are unifiable only if $\ell = \ell'$, for all $\ell'\colon d' \in D$, it must be the case that $\ell = \ell_j$ for all $j \in \{1, \ldots, i\}$. We can thus apply the induction hypothesis to obtain $\mathcal{B}_0 \in \mathit{tr}_{D\setminus\{\ell\colon d_1,\ldots,\ell\colon d_i\}}(\mathcal{A}_0)$ where $[\![M, \emptyset; \mathcal{B}_0]\!]\ \mathcal{I}$. Now pick the largest subset $\{k_1, \ldots, k_m\}$ of $\{i + 1, \ldots, n\}$ such that $\ell_{k_j} = \ell$ for all $0 \leq j \leq m$. Then $|D|_\ell = \{\ell\colon d_1, \ldots \ell\colon d_i, \ell\colon d_{k_1}, \ldots, \ell\colon d_{k_m}\}$. Let $\mathcal{B} = \ell\colon (t, s) \doteq d_1. \cdots .\ell\colon (t, s) \doteq d_i.\ell\colon (t, s) \not\doteq d_{k_1}. \cdots .\ell\colon (t, s) \not\doteq d_{k_m}.\mathcal{B}_0$. Thus $\mathcal{B} \in \mathit{tr}_D(\mathcal{A})$ and $[\![M, \emptyset; \mathcal{B}]\!]\ \mathcal{I}$ which concludes the case.

□

**Lemma 11** *Let $i \in \{1, 2\}$ be a label. If $\mathcal{B} \in \mathit{tr}_D(\mathcal{A})$ then $\mathcal{B}|_i \in \mathit{tr}_{|D|_i \cup |D|_\star}(\mathcal{A}|_i)$.*

*Proof.* The lemma follows from an induction over the structure of $\mathcal{A}$. In this sketch we will only show the $t \mathrel{\dot\in} s$ and $\mathsf{delete}(t, s)$ cases, and we will only consider the case where $i = 1$. All remaining cases are similarly proven.

- Case $\mathcal{A} = (\ell\colon t \mathrel{\dot\in} s).\mathcal{A}'$: In this case we know that $\mathcal{B}$ must be of the form $(\ell\colon (t, s) \doteq d).\mathcal{B}'$ for some $\ell\colon d \in |D|_\ell$ and $\mathcal{B}' \in \mathit{tr}_D(\mathcal{A}')$. From the induction hypothesis we can now conclude that

  $$\mathcal{B}'|_1 \in \mathit{tr}_{|D|_1 \cup |D|_\star}(\mathcal{A}'|_1) \qquad \text{(IH)}$$

  We now show that $\mathcal{B}|_1 \in \mathit{tr}_{|D|_1 \cup |D|_\star}(\mathcal{A}|_1)$ by a case analysis on the label $\ell$:
  - $\ell = \star$ or $\ell = 1$:
    In these cases we have that $\mathcal{A}|_1 = (\ell\colon t \mathrel{\dot\in} s).(\mathcal{A}'|_1)$ and $\mathcal{B}|_1 = (\ell\colon (t, s) \doteq d).(\mathcal{B}'|_1)$. We also have that $\ell\colon d \in |D|_1 \cup |D|_\star$. Thus the case follows from (IH) and the definition of $\mathit{tr}$.
  - $\ell = 2$:
    In this case we have that $\mathcal{A}|_1 = \mathcal{A}'|_1$ and $\mathcal{B}|_1 = \mathcal{B}'|_1$. Thus the case follows immediately from (IH).
- Case $\mathcal{A} = (\ell\colon \mathsf{delete}(t, s)).\mathcal{A}'$: In this case we know that $\mathcal{B}$ must be of the form $\ell\colon (t, s) \doteq d_1. \cdots .\ell\colon (t, s) \doteq d_i.\ell\colon (t, s) \not\doteq d_{i+1}. \cdots .\ell\colon (t, s) \not\doteq d_n.\mathcal{B}'$ for some $\mathcal{B}' \in \mathit{tr}_{D'}(\mathcal{A}')$ and $0 \leq i \leq n$ where $|D|_\ell = \{\ell\colon d_1, \ldots, \ell\colon d_i, \ldots, \ell\colon d_n\}$ and $D' = D \setminus \{\ell\colon d_1, \ldots, \ell\colon d_i\}$. From the induction hypothesis we can now conclude that

  $$\mathcal{B}'|_1 \in \mathit{tr}_{|D'|_1 \cup |D'|_\star}(\mathcal{A}'|_1) \qquad \text{(IH)}$$

  We now show that $\mathcal{B}|_1 \in \mathit{tr}_{|D|_1 \cup |D|_\star}(\mathcal{A}|_1)$ by a case analysis on the label $\ell$:

- $\ell = \star$ or $\ell = 1$:
  In these cases we have that $|D'|_1 \cup |D'|_\star = (|D|_1 \cup |D|_\star) \setminus \{\ell\colon d_1, \dots, \ell\colon d_i\}$ and $\mathcal{B}|_1 = (\ell\colon (t,s) \doteq d_1. \cdots .\ell\colon (t,s) \doteq d_i.\ell\colon (t,s) \not\doteq d_{i+1}. \cdots .\ell\colon (t,s) \not\doteq d_n).(\mathcal{B}'|_1)$ and $\mathcal{A}|_1 = (\ell\colon \mathsf{delete}(t,s)).(\mathcal{A}|_1)$. Thus the case follows from (IH) and the definition of $tr$.
- $\ell = 2$:
  In this case we have that $\mathcal{A}|_1 = \mathcal{A}'|_1$, $\mathcal{B}|_1 = \mathcal{B}'|_1$, $|D'|_1 = |D|_1$, and $|D'|_\star = |D|_\star$. Thus the case follows immediately from (IH).

□

Recall that the following Theorem 2 has already been proven in Isabelle/HOL for ordinary constraints. In the following we will provide a proof sketch that explains how to lift this result to stateful constraints using the reduction $tr$.

**Theorem 2.** *If $(\mathcal{A}, Sec)$ is parallel composable and $\mathcal{I} \models \mathcal{A}$ then there exists a well-typed interpretation $\mathcal{I}_\tau$ such that either $\mathcal{I}_\tau \models \mathcal{A}|_1$ and $\mathcal{I}_\tau \models \mathcal{A}|_2$ or some prefix $\mathcal{A}'$ of $\mathcal{A}$ leaks a secret from $Sec$ under $\mathcal{I}_\tau$.*

*Proof.* From the assumptions, Lemma 9, and Lemma 10 we can obtain a parallel composable $\mathcal{B}$ such that

$$\mathcal{B} \in tr(\mathcal{A}) \text{ and } \mathcal{I} \models \mathcal{B} \tag{*}$$

From the Isabelle/HOL-formalized version of the theorem we can then obtain a well-typed interpretation $\mathcal{I}_\tau$ such that either

1. $\mathcal{I}_\tau \models \mathcal{B}|_1$ and $\mathcal{I}_\tau \models \mathcal{B}|_2$, or
2. some prefix $\mathcal{B}'$ of $\mathcal{B}$ leaks a secret from $Sec$ under $\mathcal{I}_\tau$.

In the former case it follows from Lemma 11, Lemma 10, and (*) that $\mathcal{I}_\tau \models \mathcal{A}|_1$ and $\mathcal{I}_\tau \models \mathcal{A}|_2$ (note that the assumption of Lemma 10 follows from the fact that $\mathcal{B}$ is parallel composable and that the assumption is also preserved during projections). In the latter case we can obtain a secret $s \in Sec \setminus declassified(\mathcal{B}', \mathcal{I}_\tau)$ such that either $\mathcal{I}_\tau \models \mathcal{B}'|_1.\mathsf{send}(s)$ or $\mathcal{I}_\tau \models \mathcal{B}'|_2.\mathsf{send}(s)$. We need to prove that some prefix of $\mathcal{A}$ leaks the secret $s$ and we will do so using the semantic equivalence of $tr$. However, there is not necessarily a corresponding prefix of $\mathcal{A}$ with $\mathcal{B}'$ as a translation, and we need such a prefix to apply Lemma 10. Therefore we consider the longest prefix $\mathcal{B}''$ of $\mathcal{B}'$ that ends in a $\mathsf{receive}$ step (which must exist if $s$ is not derivable from the empty intruder knowledge). For $\mathcal{B}''$ we can prove that there exists some prefix $\mathcal{A}''$ of $\mathcal{A}$ such that $\mathcal{B}'' \in tr(\mathcal{A}'')$. We also know that either $\mathcal{I}_\tau \models \mathcal{B}''|_1.\mathsf{send}(s)$ or $\mathcal{I}_\tau \models \mathcal{B}''|_2.\mathsf{send}(s)$ because $\mathcal{B}'$ and $\mathcal{B}''$ have the same intruder knowledges (also after projections). Moreover, $declassified(\mathcal{B}'', \mathcal{I}_\tau) = declassified(\mathcal{A}'', \mathcal{I}_\tau)$ and $ik(\mathcal{B}'') = ik(\mathcal{A}'')$ (also after projections). Thus we have that $\mathcal{A}''$ leaks a secret from $Sec$ under $\mathcal{I}_\tau$ and we can therefore conclude the proof. □

Now that we have proven the result on the constraint level we can now prove Theorem 3 for stateful protocols. The main idea is to prove the result for $\mathfrak{P}^\bullet$ (Lemma 14) from which the theorem follows.

**Lemma 12** $\mathit{traces}(\mathcal{P}_1 \parallel \mathcal{P}_2) \subseteq \mathfrak{P}^\bullet$

*Proof.* A constraint $\mathcal{A} \in \mathit{traces}(\mathcal{P}_1 \parallel \mathcal{P}_2)$ consists of an interleaving of two reachable constraints $\mathcal{A}_1 \in \mathit{traces}(\mathcal{P}_1)$ and $\mathcal{A}_2 \in \mathit{traces}(\mathcal{P}_2)$. Consider $\mathcal{A}|_1$. We need to prove that this constraint is in $\mathit{traces}(\mathcal{P}_1^\bullet)$. We have that $\mathcal{A}|_1$ consists of an interleaving of $\mathcal{A}_1|_1$ and $\mathcal{A}_2|_1$, and that $\mathcal{A}_1|_1 = \mathcal{A}_1 \in \mathit{traces}(\mathcal{P}_1)$ and $\mathcal{A}_2|_1 = \mathcal{A}_2|_\star \in \mathit{traces}(\mathcal{P}_2^\star)$. Thus $\mathcal{A}|_1 \in \mathit{traces}(\mathcal{P}_1^\bullet)$ because $\mathcal{P}_1^\bullet = \mathcal{P}_1 \cup \mathcal{P}_2^\star$. By a similar argument we can prove that $\mathcal{A}|_2 \in \mathit{traces}(\mathcal{P}_2^\bullet)$. □

**Lemma 13** *$(\mathcal{P}_1 \parallel \mathcal{P}_2, \mathit{Sec})$ is parallel composable if and only if $(\mathfrak{P}^\bullet, \mathit{Sec})$ is parallel composable.*

*Proof.* Note that all constraint steps that occur in $\mathit{traces}(\mathcal{P}_1 \parallel \mathcal{P}_2)$ also occur in $\mathfrak{P}^\bullet$, and vice versa. Since all but our well-formedness requirements universally quantifies over the terms and steps occurring in the protocols we have that these requirements are satisfied for $(\mathcal{P}_1 \parallel \mathcal{P}_2, \mathit{Sec})$ if and only if they are satisfied for $(\mathfrak{P}^\bullet, \mathit{Sec})$. For the well-formedness requirements note that we require all the reachable constraints plus all of the projections to be well-formed. Since $\mathfrak{P}^\bullet$ really only differs from $\mathit{traces}(\mathcal{P}_1 \parallel \mathcal{P}_2)$ by including $\star$-projections of (and interleavings of) constraints from $\mathit{traces}(\mathcal{P}_1 \parallel \mathcal{P}_2)$ we have that the well-formedness requirements for $\mathit{traces}(\mathcal{P}_1 \parallel \mathcal{P}_2)$ are satisfied if and only if they are satisfied for $\mathfrak{P}^\bullet$. □

**Lemma 14** *If $(\mathfrak{P}^\bullet, \mathit{Sec})$ is parallel composable, and $\mathcal{P}_1^\bullet$ is well-typed secure in isolation, then for any attack $\mathcal{A}.(1\colon \mathsf{send}(\mathsf{attack}_1)) \in \mathfrak{P}^\bullet$ on $\mathcal{P}_1$, there exists some prefix $\mathcal{A}' \in \mathit{traces}(\mathcal{P}_2^\bullet)$ of $\mathcal{A}|_2$ that leaks a secret under a well-typed model.*

*Proof.* We have proven this theorem in Isabelle/HOL for stateless protocols. The proof of the stateful version is actually similar to the proof of the stateless one. We first prove that any $\mathcal{A}.(1\colon \mathsf{send}(\mathsf{attack}_1)) \in \mathfrak{P}^\bullet$ is parallel composable. Then we can apply Theorem 2 since the constraint is satisfiable (otherwise it would not be an attack), and since $\mathcal{P}_1^\bullet$ is secure it must be the case that some prefix of $\mathcal{A}' \in \mathit{traces}(\mathcal{P}_2^\bullet)$ of $\mathcal{A}|_2$ leaks a secret. □

From Lemma 12, 13, and 14 follows our main theorem:

**Theorem 3.** *If $(\mathcal{P}_1 \parallel \mathcal{P}_2, \mathit{Sec})$ is parallel composable, $\mathcal{P}_1 \parallel \mathcal{P}_2^\star$ is well-typed secure in isolation, and $\mathcal{P}_1^\star \parallel \mathcal{P}_2$ does not leak a secret under any well-typed model, then all goals of $\mathcal{P}_1$ hold in $\mathcal{P}_1 \parallel \mathcal{P}_2$ (even in the untyped model).*

As a consequence of Theorem 3 we have the following corollary:

**Corollary 1.** *If $(\mathcal{P}_1 \parallel \mathcal{P}_2, \mathit{Sec})$ is parallel composable and $\mathcal{P}_1 \parallel \mathcal{P}_2^\star$ and $\mathcal{P}_1^\star \parallel \mathcal{P}_2$ are both secure in isolation then the composition $\mathcal{P}_1 \parallel \mathcal{P}_2$ is also secure (even in the untyped model).*

*Proof.* Apply Theorem 3 twice: once to $\mathcal{P}_1^\bullet$ and once to $\mathcal{P}_2^\bullet$.