# Typing and Compositionality for Security Protocols: A Generalization to the Geometric Fragment (Extended Version)

Omar Almousa[1], Sebastian Mödersheim[1], Paolo Modesti[2], and Luca Viganò[3]

[1] DTU Compute, Lyngby, Denmark
[2] School of Computing Science, Newcastle University, UK
[3] Department of Informatics, King's College London, UK

**Abstract.** We integrate, and improve upon, prior relative soundness results of two kinds. The first kind are typing results showing that any security protocol that fulfils a number of sufficient conditions has an attack if it has a well-typed attack. The second kind considers the parallel composition of protocols, showing that when running two protocols in parallel allows for an attack, then at least one of the protocols has an attack in isolation. The most important generalization over previous work is the support for all security properties of the geometric fragment.

## 1 Introduction

**Context and motivation** Relative soundness results have proved helpful in the automated verification of security protocols as they allow for the reduction of a complex verification problem into a simpler one, if the protocol in question satisfies sufficient conditions. These conditions are of a syntactic nature, i.e., can be established without an exploration of the state space of the protocol.

A first kind of such results are *typing results* [11, 4, 16, 2]. In this paper, we consider a *typed model*, a restriction of the standard protocol model in which honest agents do not accept any ill-typed messages. This may seem unreasonable at first sight, since in the real-world agents have no way to tell the type of a random bitstring, let alone distinguish it from the result of a cryptographic operation; yet in the model, they "magically" accept only well-typed messages. The relative soundness of such a typed model means that if the protocol has an attack, then it also has a well-typed attack. This does not mean that the intruder cannot send ill-typed messages, but rather that this does not give him any advantage as he could perform a "similar" attack with only well-typed messages. Thus, if we are able to verify that a protocol is secure in the typed model, then it is secure also in an untyped model. Typically, the conditions sufficient to achieve such a result are that all composed message patterns of the protocol have a different (intended) type that can somehow be distinguished, e.g., by a tag. The restriction to a typed model in some cases yields a decidable verification problem, allows for the application of more tools and often significantly reduces verification time in practice [4, 3].

A similar kind of relative soundness results appears in *compositional reasoning*. We consider in this paper the *parallel composition* of protocols, i.e., running two protocols over the same communication medium, and these protocols may use, e.g., the same long-term public keys. (In the case of disjoint cryptographic material, compositional reasoning is relatively straightforward.) The compositionality result means to show that if two protocols satisfy their security goals in isolation, then their parallel composition is secure, provided the protocols meet certain sufficient conditions. Thus, it suffices to verify the protocols in isolation. The sufficient conditions in this case are similar to the typing result: every composed message can be uniquely attributed to one of the two protocols, which again may be achieved, e.g., by tags.

**Contributions** The contributions of this paper are thus twofold. First, we unify and thereby simplify existing typing and compositionality results: we recast them as an instance of the same basic principle and of the same proof technique. In a nutshell, this technique is to reduce the search for attacks to solving constraint reduction in a symbolic model. For protocols that satisfy the respective sufficient conditions, the constraint reduction will never make an ill-typed substitution, where for compositionality "ill-typed" means to unify protocol messages from two different protocols.

Second, this systematic approach also allows us to significantly generalize existing results to a larger set of protocols and security properties. For what concerns protocols, our soundness results do not require a particular fixed tagging scheme like most previous works, but use more liberal requirements that are satisfied by many existing real-world protocols like TLS.

While many existing results are limited to simple secrecy goals, we prove our results for the entire geometric fragment suggested by Guttman [9]. We even augment this fragment with the ability to directly refer to the intruder knowledge in the antecedent of goals; while this does not increase expressiveness, it is very convenient in specifications. In fact, handling the geometric fragment also constitutes a slight generalization of existing constraint-reduction approaches.

We proceed as follows. In § 2 and § 3, we introduce a symbolic protocol model based on strands and properties in the geometric fragment. In § 4, we reduce verification of the security properties to solving constraints. In § 5 and § 6, we give our typing and parallel compositionality results. In § 7, we introduce a tool that automatically checks if protocols are parallel-composable and report about first experimental results. In § 8, we conclude and discuss the related work. Proofs of the technical results are given in the appendix.

## 2 Messages, Formats and the Intruder Model

### 2.1 Messages

Let $\Sigma$ be a finite set of *operators* (also referred to as *function symbols*); as a concrete example, Table 1 shows a $\Sigma$ that is representative for a wide range of security protocols. Further, let $\mathcal{C}$ be a countable set of *constants* and $\mathcal{V}$ a countable set of *variables*, such that $\Sigma$, $\mathcal{V}$ and $\mathcal{C}$ are pairwise disjoint. We write $\mathcal{T}_{\Sigma\cup\mathcal{C}}(\mathcal{V})$ for the set of *terms* built with these constants, variables and operators, and $\mathcal{T}_{\Sigma\cup\mathcal{C}}$ for the set of *ground terms*. We call a term $t$ *atomic* (and write *atomic*$(t)$) if $t \in \mathcal{V} \cup \mathcal{C}$, and *composed* otherwise. We use also other standard notions such as *subterm*, denoted by $\sqsubseteq$, and *substitution*, denoted by $\sigma$.

The set of constants $\mathcal{C}$ is partitioned into three countable and pairwise disjoint subsets: (i) the set $\mathcal{C}_{P_i}$ of *short-term constants* for each protocol $P_i$, denoting the constants that honest agents freshly generate in $P_i$; (ii) the set $\mathcal{C}_{priv}$ of *long-term secret constants*; and (iii) the set $\mathcal{C}_{pub}$ of *long-term public constants*. This partitioning will be useful for compositional reasoning: roughly speaking, we will allow the intruder to obtain all public constants, and define that it is an attack if the intruder finds out any of the secret constants.

### 2.2 Formats

We use in this paper a notion of *formats* that is crucial to make our typing and compositionality results applicable to real-world protocols like TLS. Here, we break with the formal-methods tradition of representing clear-text structures of data by a *pair* operator $(\cdot,\cdot)$. For instance, a typical specification may contain expressions like $(A, NA)$ and $(NB, (KB, A))$. This representation neglects the details of a protocol implementation that may employ various mechanisms to enable a receiver to decompose a message in a unique way (e.g., field-lengths or XML-tags). The abstraction has the disadvantage that it may easily lead to false positives and false negatives. For example, the two messages above have a unifier $A \mapsto NB$ and $NA \mapsto (KB, NA)$, meaning that a message meant as $(A, NA)$ may accidentally be parsed as $(NB, (KB, A))$, which could lead to a "type-flaw" attack. This attack may, however, be completely unrealistic in reality.

To handle this, previous typing results have used particular *tagging schemes*, e.g., requiring that each message field starts with a tag identifying the type of that field. Similarly, compositionality results have often required that each encrypted message of a protocol starts with a tag identifying the protocol that this message was meant for. Besides the fact that this does not really solve the problem of false positives and false negatives due to the abstraction, practically no existing protocol uses exactly this schema. Moreover, it is completely unrealistic to think that a widely used protocol like TLS would be changed just to make it compatible with the assumptions of an academic paper — the only chance to have it changed is to point out a vulnerability that can be fixed by the change.

Formats are a means to have a faithful yet abstract model. We define formats as functions from data-packets to concrete strings. For example, a format from TLS is **client_hello(time, random, session_id, cipher_suites, comp_methods)** $=$ $\mathsf{byte}(1) \cdot \mathsf{off}_3(\mathsf{byte}(3) \cdot \mathsf{byte}(3) \cdot time \cdot random \cdot \mathsf{off}_1(session\_id) \cdot \mathsf{off}_2(cipher\_suites) \cdot \mathsf{off}_1(comp\_methods))$, where $\mathsf{byte}(n)$ means one concrete byte of value $n$, $\mathsf{off}_k(m)$ means that $m$ is a message of variable length followed by a field of $k$ bytes, and $\cdot$ represents string concatenation.

| Description | Operator | Analysis rule |
|---|---|---|
| Symmetric encryption | $\mathsf{scrypt}(\cdot,\cdot)$ | $\mathtt{Ana}(\mathsf{scrypt}(k,m)) = (\{k\},\{m\})$ |
| Asymmetric encryption | $\mathsf{crypt}(\cdot,\cdot)$ | $\mathtt{Ana}(\mathsf{crypt}(\mathsf{pub}(t),m)) = (\{t\},\{m\})$ |
| Signature | $\mathsf{sign}(\cdot,\cdot)$ | $\mathtt{Ana}(\mathsf{sign}(t,m)) = (\{\emptyset\},\{m\})$ |
| Formats, e.g., $\mathsf{f}_1$ | $\mathsf{f}_1(t_1,\cdots,t_n)$ | $\mathtt{Ana}(\mathsf{f}_1(t_1,\cdots,t_n)) = (\emptyset,\{t_1,\cdots,t_n\})$ |
| One-way functions, e.g., $\mathsf{hash}$ | $\mathsf{hash}(\cdot)$ | $\mathtt{Ana}(\mathsf{hash}(t)) = (\{\emptyset\},\{\emptyset\})$ |
| Public key of a given private key | $\mathsf{pub}(\cdot)$ | $\mathtt{Ana}(\mathsf{pub}(t)) = (\{\emptyset\},\{\emptyset\})$ |
| All other terms | | $\mathtt{Ana}(t) = (\{\emptyset\},\{\emptyset\})$ |

**Table 1.** Example Operators $\Sigma$

In the abstract model, we are going to use only abstract terms like the part in bold in the above example. It is shown in [17] that under certain conditions (on formats) this is a sound abstraction, i.e., without false positives or false negatives. These conditions are basically that formats must be parsed in an unambiguous way and must be pairwise disjoint, which means that we can treat formats just as free function symbols in the formal model. Both in typing and compositionality, these conditions allow us to apply our results to real world protocols no matter what formatting scheme they actually use (e.g., a TLS message cannot be accidentally be parsed as an EAC message). In fact, these reasonable conditions are satisfied by many protocols in practice, and whenever they are violated, typically we have a good chance to find actual vulnerabilities.

We will model formats as *transparent* in the sense that if the intruder learns $\mathsf{f}(t_1,\ldots,t_n)$, then he also obtains the $t_i$.

### 2.3 Intruder knowledge and deduction rules

We specify how the intruder can compose and decompose messages in the style of the Dolev-Yao intruder model.

**Definition 1.** *An* intruder knowledge $M$ *is a finite set of ground terms* $t \in \mathcal{T}_{\Sigma\cup\mathcal{C}}$*. Let* $\mathtt{Ana}(t) = (K,T)$ *be a function that returns for every term* $t$ *a pair* $(K,T)$ *of finite sets of subterms of* $t$*. We define* $\vdash$ *to be the least relation between a knowledge* $M$ *and a term* $t$ *that satisfies the following* intruder deduction rules*:*

$$\frac{}{M \vdash t}\ \text{(Axiom)},\ t \in M \qquad \frac{}{M \vdash c}\ \text{(Public)},\ c \in \mathcal{C}_{pub} \qquad \frac{M \vdash t_1 \quad \cdots \quad M \vdash t_n}{M \vdash f(t_1,\cdots,t_n)}\ \text{(Compose)},\ f \in \Sigma^n$$

$$\frac{M \vdash t \quad M \vdash k_1 \quad \cdots \quad M \vdash k_n}{M \vdash t_i}\ \text{(Decompose)},\ \mathtt{Ana}(t) = (K,T),\ K = \{k_1,\cdots,k_n\},\ t_i \in T$$

The rules *(Axiom)* and *(Public)* formalize that the intruder can derive any term $t \in M$ that is in his knowledge and every long-term public constant $c \in \mathcal{C}_{pub}$, respectively, and the *(Compose)* rule formalizes that he can use compose known terms with any operator in $\Sigma$ (where $n$ denotes the arity of $f$). Table 1 provides an example $\Sigma$ for standard cryptographic operators, along with the $\mathtt{Ana}$ function defined for each of them, which are available to all agents, including the intruder.

For message decomposition, we namely rely on analysis rules for terms in the form of $\mathtt{Ana}(t) = (K, T)$, which intuitively says that if the intruder knows the keys in set $K$, then he can analyze the term $t$ and obtain the set of messages $T$. We require that all elements of $K$ and $T$ are subterms of $t$ (without any restriction, the relation $\vdash$ would be undecidable). Consider, e.g., the analysis rule for symmetric encryption given in Table 1: $\mathtt{Ana}(\mathsf{scrypt}(k, m)) = (\{k\}, \{m\})$ says that given a term $\mathsf{scrypt}(k, m)$ one needs the key $\{k\}$ to derive $\{m\}$. By default, atomic terms cannot be analyzed, i.e., $\mathtt{Ana}(t) = (\emptyset, \emptyset)$. The generic *(Decompose)* deduction rule then formalizes that for any term with an $\mathtt{Ana}$ rule, if the intruder can derive the keys in $K$, he can also derive all the subterms of $t$ in $T$.

## 3 Protocol Semantics

We define the following notions. A protocol consists of a set of *operational strands* (an extension of the strands of [10]) and a set of *goal predicates* that the protocol is supposed to achieve. The semantics of a protocol is an infinite-state transition system over symbolic states and security means that all reachable states satisfy the goal predicates. A *symbolic state* $(\mathcal{S}; M; E; \phi)$ consists of a set $\mathcal{S}$ of operational strands (representing the honest agents), the intruder knowledge $M$, a set $E$ of events that have occurred, and a symbolic constraint $\phi$ on the free variables occurring in the state. We first define constraints, then operational strands, the transition relation on symbolic states, and finally the goal predicates.

### 3.1 Symbolic Constraints

The syntax of *symbolic constraints* is

$$\phi \ := \ M \vdash t \mid \phi_\sigma \mid \neg\exists\bar{x}.\, \phi_\sigma \mid \phi \wedge \phi \mid \underbrace{\phi \vee \phi \mid \exists\bar{x}.\, \phi}_{\star} \quad \text{with} \quad \phi_\sigma \ := \ s \doteq t \mid \phi_\sigma \wedge \phi_\sigma$$

where $s, t$ range over terms in $\mathcal{T}_{\Sigma\cup\mathcal{C}}(\mathcal{V})$, $M$ is a finite set of terms (not necessarily ground) and $\bar{x}$ is list of variables. The sublanguage $\phi_\sigma$ defines *equations* on messages, and we can existentially quantify variables in them, e.g., consider a $\phi$ of the form $\exists x.\, y \doteq f(x)$. We refer to equations also as *substitutions* since the application of the standard most general unifier on a conjunction of equations results in a set of substitutions. The constraints can contain such substitutions in positive and negative form (excluding all instances of a particular substitution).

$M \vdash t$ is an *intruder constraint*: the intruder must be able to derive term $t$ from knowledge $M$. Note that we have no negation at this level, i.e., we cannot write negated intruder constraints. A *base constraint* is a constraint built according to this grammar without the two cases marked $\star$, i.e., disjunction $\phi \vee \phi$ and existential quantification $\exists\bar{x}.\, \phi$, which may only occur in negative substitutions.

For ease of writing, we define the semantics of the constraint language as standard for each construct (rather than following strictly the grammar of $\phi$).

**Definition 2.** *Given an interpretation $\mathcal{I}$, which maps each variable in $\mathcal{V}$ to a ground term in $\mathcal{T}_\Sigma$, and a symbolic constraint $\phi$, the model relation $\mathcal{I} \models \phi$ is:*

$$\mathcal{I} \models M \vdash t \;\text{ iff }\; \mathcal{I}(M) \vdash \mathcal{I}(t) \qquad \mathcal{I} \models s \doteq t \;\text{ iff }\; \mathcal{I}(s) = \mathcal{I}(t) \qquad \mathcal{I} \models \neg\phi \;\text{ iff }\; \text{not } \mathcal{I} \models \phi$$
$$\mathcal{I} \models \phi_1 \wedge \phi_2 \;\text{ iff }\; \mathcal{I} \models \phi_1 \text{ and } \mathcal{I} \models \phi_2 \qquad \mathcal{I} \models \phi_1 \vee \phi_2 \;\text{ iff }\; \mathcal{I} \models \phi_1 \text{ or } \mathcal{I} \models \phi_2$$
$$\mathcal{I} \models \exists x.\phi \;\text{ iff }\; \text{there is a term } t \in \mathcal{T}_\Sigma \text{ such that } \mathcal{I}[x \mapsto t] \models \phi$$

*We say that $\mathcal{I}$ is a* model *of $\phi$ iff $\mathcal{I} \models \phi$, and that $\phi$ is* satisfiable *iff it has a model. Two constraints are* equivalent*, denoted by $\equiv$, iff they have the same models. We define as standard the* variables*, denoted by $var(\cdot)$, and the* free variables*, denoted by $fv(\cdot)$, of terms, sets of terms, equations, and constraints. A constraint $\phi$ is* closed*, in symbols $closed(\phi)$, iff $fv(\phi) = \emptyset$.*

Every constraint $\phi$ can be quite straightforwardly transformed into an equivalent constraint of the form

$$\phi \;\equiv\; \exists \bar{x}.\, \phi_1 \vee \ldots \vee \phi_n$$

where the $\phi_i$ are base constraints. Unless noted otherwise, in the following we will assume that constraints are in this form.

**Definition 3.** *A constraint is* well-formed *if each of its base constraints $\phi_i$ satisfies the following condition: we can order the conjuncts of $\phi_i$ such that $\phi_i = M_1 \vdash t_1 \wedge \ldots \wedge M_n \vdash t_n \wedge \phi'_i$, where $\phi'_i$ contains no further $\vdash$ constraints and such that $M_j \subseteq M_{j+1}$ (for $1 \leq j < n$) and $fv(M_j) \subseteq fv(t_1) \cup \ldots \cup fv(t_{j-1})$.*

Intuitively, this condition expresses that the intruder knowledge grows monotonically and all variables that occur in an intruder knowledge occur in a term that the intruder sent earlier in the protocol execution. We will ensure that all constraints that we deal with are well-formed.

### 3.2 Operational Strands

In the original definition of [19], a strand denotes a part of a concrete protocol execution, namely, a sequence of ground messages that an agent sends and receives. We introduce here an extension that we call *operational strands*, where terms may contain variables, there may be positive and negative equations on messages, and agents may create events over which we can formulate the goals:

$$S \;:=\; \mathsf{send}(t).S \mid \mathsf{receive}(t).S \mid \mathsf{event}(t).S \mid (\exists \bar{x}.\, \phi_\sigma).S \mid (\neg\exists \bar{x}.\phi_\sigma).S \mid 0$$

where $\phi_\sigma$ is as defined above; we will omit the parentheses when there is no risk of confusing the dots. *fv* and *closed* extend to operational strands as expected, with the exception of the receiving step, which can bind variables: we set $fv(\mathsf{receive}(t).S) = fv(S) \setminus fv(t)$. According to the semantics that we define below, in $\mathsf{receive}(x).\mathsf{receive}(f(x)).\mathsf{send}(x).0$ the variable $x$ is bound actually in the first receive, i.e., the strand is equivalent to $\mathsf{receive}(x).\mathsf{receive}(y).(y \doteq f(x)).\mathsf{send}(x).0\,.$

A *symbolic state* $(\mathcal{S}; M; E; \phi)$ consists of a (finite or countable) set $\mathcal{S}$ of closed operational strands, a finite set[4] $M$ of terms representing the intruder knowledge, a finite set $E$ of events, and a formula $\phi$. $fv(\cdot)$ and *closed* extend to symbolic states again as expected. We ensure that $fv(\mathcal{S}) \cup fv(M) \cup fv(E) \subseteq fv(\phi)$ for all reachable states $(\mathcal{S}; M; E; \phi)$, and that $\phi$ is well-formed. This is so since in the transition system shown shortly, the operational strands of the initial state are closed and the transition relation only adds new variables in the case of $\mathsf{receive}(t)$, but in this case $\phi$ is updated with $M \vdash t$.

A *protocol specification* $(\mathcal{S}_0, G)$ (or simply *protocol*) consists of a set $\mathcal{S}_0$ of closed operational strands and a set $G$ of goal predicates (defined below). For simplicity, we assume that the bound variables of any two different strands in $\mathcal{S}_0$ are disjoint (which can be achieved by $\alpha$-renaming). The strands in $\mathcal{S}_0$ induce an *infinite-state transition system* with *initial state* $(\mathcal{S}_0; \emptyset; \emptyset; \top)$ and a transition relation $\Rightarrow$ defined as the least relation closed under six transition rules:

T1 $(\{\mathsf{send}(t).S\} \cup \mathcal{S}; M; E; \phi) \Rightarrow (\{S\} \cup \mathcal{S}; M \cup \{t\}; E; \phi)$
T2 $(\{\mathsf{receive}(t).S\} \cup \mathcal{S}; M; E; \phi) \Rightarrow (\{S\} \cup \mathcal{S}; M; E; \phi \wedge M \vdash t)$
T3 $(\{\mathsf{event}(t).S\} \cup \mathcal{S}; M; E; \phi) \Rightarrow (\{S\} \cup \mathcal{S}; M; E \cup \mathsf{event}(t); \phi)$
T4 $(\{\phi'.S\} \cup \mathcal{S}; M; E; \phi) \Rightarrow (\{S\} \cup \mathcal{S}; M; E; \phi \wedge \phi')$
T5 $(\{0\} \cup \mathcal{S}; M; E; \phi) \Rightarrow (\mathcal{S}; M; E; \phi)$
T6 $(\mathcal{S}; M; E; \phi) \Rightarrow (\mathcal{S}; M; E \cup \{lts(c)\}; \phi)$ for every $c \in \mathcal{C}_{priv}$

The rule T1 formalizes that sent messages are added to the intruder knowledge $M$. T2 formalizes that an honest agent receives a message of the form $t$, and that the intruder must be able to create that message from his current knowledge, expressed by the new constraint $M \vdash t$; this indirectly binds the free variables of the rest of the strand in the sense that they are now governed by the constraints of the state. (In a non-symbolic model, one would at this point instead need to consider all ground instances of $t$ that the intruder can generate.) T3 formalizes that we add events to the set $E$. T4 simply adds the constraint $\phi'$ to the constraint $\phi$. T5 says that if a strand reaches $\{0\}$, then we remove it. Finally, for every secret constant $c$ in $\mathcal{C}_{priv}$, T6 adds the event $lts(c)$ to the set $E$. (We define later as a goal that the intruder never obtains any $c$ for which $lts(c) \in E$.) We cannot have this in the initial set $E$ as we need it to be finite; this construction is later crucial in the parallel composition proof as we can at any time blame a protocol (in isolation) that leaks a secret constant. We discuss below that in practice this semantic rule does not cause trouble to the verification of the individual protocols.

### 3.3 Goal Predicates in the Geometric Fragment

We express goals by *state formulas in the geometric fragment* [9]. Here, we also allow to directly talk about the intruder knowledge, but in a restricted manner so that we obtain constraints of the form $\phi$. Security then means: every reachable

[4] Some approaches instead use multi-sets as we may have several identical strands, but since one can always make a strand unique, using sets is without loss of generality.

state in the transition system induced by $\mathcal{S}_0$ satisfies each state formula, and thus an *attack* is a reachable state where at least one goal does not hold.

The constraints $\phi$ we have defined above are interpreted only with respect to an interpretation of the free variables, whereas the state formulas are evaluated with respect to a symbolic state, including the current intruder knowledge and events that have occurred (as before, we define the semantics for each construct).

**Definition 4.** State formulas $\Psi$ in the geometric fragment *are defined as:*

$$\Psi := \forall \bar{x}.\,(\psi \implies \psi_0) \text{ with } \begin{cases} \psi := \mathsf{ik}(t) \mid \mathsf{event}(t) \mid t \doteq t' \mid \psi \wedge \psi' \mid \psi \vee \psi' \mid \exists \bar{x}.\psi \\ \psi_0 := \mathsf{event}(t) \mid t \doteq t' \mid \psi_0 \wedge \psi_0' \mid \psi_0 \vee \psi_0' \mid \exists \bar{x}.\psi_0 \end{cases}$$

*where* $\mathsf{ik}(t)$ *denotes that the intruder knows the term* $t$*.* $fv(\cdot)$ *and closed extend to state formulas as expected. Given a state formula* $\Psi$*, an interpretation* $\mathcal{I}$*, and a state* $\mathfrak{S} = (\mathcal{S}; M; E; \phi)$*, we define* $\mathcal{I}, M, E \models_{\mathfrak{S}} \Psi$ *as:*

$$\begin{array}{ll}
\mathcal{I}, M, E \models_{\mathfrak{S}} \mathsf{event}(t) & \text{iff } \mathcal{I}(\mathsf{event}(t)) \in \mathcal{I}(E) \\
\mathcal{I}, M, E \models_{\mathfrak{S}} \mathsf{ik}(t) & \text{iff } \mathcal{I}(M) \vdash \mathcal{I}(t) \\
\mathcal{I}, M, E \models_{\mathfrak{S}} s \doteq t & \text{iff } \mathcal{I}(s) = \mathcal{I}(t) \\
\mathcal{I}, M, E \models_{\mathfrak{S}} \Psi \wedge \Psi' & \text{iff } \mathcal{I}, M, E \models_{\mathfrak{S}} \Psi \text{ and } \mathcal{I}, M, E \models_{\mathfrak{S}} \Psi' \\
\mathcal{I}, M, E \models_{\mathfrak{S}} \Psi \vee \Psi' & \text{iff } \mathcal{I}, M, E \models_{\mathfrak{S}} \Psi \text{ or } \mathcal{I}, M, E \models_{\mathfrak{S}} \Psi' \\
\mathcal{I}, M, E \models_{\mathfrak{S}} \neg\Psi & \text{iff not } \mathcal{I}, M, E \models \Psi \\
\mathcal{I}, M, E \models_{\mathfrak{S}} \exists x.\,\Psi & \text{iff there exists } t \in \mathcal{T}_\Sigma \text{ and } \mathcal{I}[x \mapsto t] \models_{\mathfrak{S}} \Psi
\end{array}$$

**Definition 5.** *A protocol* $P = (\mathcal{S}_0, \{\Psi_0, \cdots, \Psi_n\})$*, where the* $\Psi_i$ *are closed state formulas,* has an attack against goal $\Psi_i$ *iff there exist a reachable state* $\mathfrak{S} = (\mathcal{S}; M; E; \phi)$ *in the transition system induced by* $\mathcal{S}_0$ *and an interpretation* $\mathcal{I}$ *such that* $\mathcal{I}, M, E \models_{\mathfrak{S}} \neg\Psi_i$ *and* $\mathcal{I} \models \phi$*. We also call* $\mathfrak{S}$ *an* attack state *in this case.*

Note that in this definition the interpretation $\mathcal{I}$ does not matter in $\mathcal{I}, M, E \models_{\mathfrak{S}} \neg\Psi_i$ because $\Psi_i$ is closed.

*Example 1.* If a protocol generates the event[5] $\mathsf{secret}(x_A, x_B, x_m)$ to denote that the message $x_m$ is supposed to be a secret between agents $x_A$ and $x_B$, and—optionally—the event $\mathsf{release}(x_m)$ to denote that $x_m$ is no longer a secret, then we can formalize *secrecy* via the state formula $\forall x_A x_B x_m.(\mathsf{secret}(x_A, x_B, x_m) \wedge \mathsf{ik}(x_m) \implies x_A = \mathsf{i} \vee x_B = \mathsf{i} \vee \mathsf{release}(x_m))$, where $\mathsf{i}$ denotes the intruder. The release event can be used to model declassification of secrets as needed to verify perfect forward secrecy (when other data should remain secret even under the release of temporary secrets). We note that previous compositionality approaches do not support such goals.

A typical formulation of *non-injective agreement* [13] uses the two events $\mathsf{commit}(x_A, x_B, x_m)$, which represents that $x_A$ intends to send message $x_m$ to $x_B$), and $\mathsf{running}(x_A, x_B, x_m, x_C)$, which represents that $x_B$ believes to have received $x_m$ from $x_A$, with $x_C$ a unique identifier: $\forall x_A x_B x_m x_C.\,(\mathsf{running}(x_A, x_B, x_m, x_C) \implies \mathsf{commit}(x_A, x_B, x_m) \vee x_A = \mathsf{i} \vee x_B = \mathsf{i})$, and *injective agreement* would additionally require: $\forall x_A x_B x_m x_C x_C'.\, \mathsf{running}(x_A, x_B, x_m, x_C) \wedge \mathsf{running}(x_A, x_B, x_m, x_C') \implies x_A = \mathsf{i} \vee x_B = \mathsf{i} \vee x_C = x_C'$. □

[5] Strictly speaking, we should write $\mathsf{event}(\mathsf{secret}(x_A, x_B, x_m))$ but, for readability, here and below we will omit the outer $\mathsf{event}(\cdot)$ when it is clear from context.

## 4 Constraint Solving

We first show how to translate every state formula $\Psi$ in the geometric fragment for a given symbolic state $\mathfrak{S} = (\mathcal{S}; M; E; \phi)$ into a constraint $\phi'$ (in the fragment defined in Section 3.1) so that the models of $\phi \wedge \phi'$ represent exactly all concrete instances of $\mathfrak{S}$ that violate $\Psi$. Then, we extend a rule-based procedure to solve $\phi$-style constraints (getting them into an equivalent simple form). This procedure provides the basis for our typing and parallel composition results.

### 4.1 From geometric fragment to symbolic constraints

Consider a reachable symbolic state $(\mathcal{S}; M; E; \phi)$ and a goal formula $\Psi$. As mentioned earlier, we require that $\Psi$ is closed. Let us further assume that the bound variables of $\Psi$ are disjoint from the variables (bound or free) of $\mathcal{S}$, $M$, $E$, and $\phi$. We now define a *translation function* $tr_{M,E}(\Psi) = \phi'$ where $\phi'$ represents the negation of $\Psi$ with respect to intruder knowledge $M$ and events $E$. The negation is actually manifested in the first line of the definition:

$$\begin{aligned}
&tr_{M,E}(\forall \bar{x}.\, \psi \Rightarrow \psi_0) &&= \exists \bar{x}.\, tr'_{M,E}(\psi) \wedge tr'_{M,E}(\neg\psi_0)\\
&tr'_{M,E}(\mathsf{ik}(t)) &&= M \vdash t\\
&tr'_{M,E}(\mathsf{event}(t)) &&= \textstyle\bigvee_{\mathsf{event}(s)\in E} s \doteq t\\
&tr'_{M,E}(s \doteq t) &&= s \doteq t\\
&tr'_{M,E}(\psi_1 \vee \psi_2) &&= tr'_{M,E}(\psi_1) \vee tr'_{M,E}(\psi_2)\\
&tr'_{M,E}(\psi_1 \wedge \psi_2) &&= tr'_{M,E}(\psi_1) \wedge tr'_{M,E}(\psi_2)\\
&tr'_{M,E}(\exists \bar{x}.\psi) &&= \exists \bar{x}. tr'_{M,E}(\psi)\\
&tr'_{M,E}(\neg\mathsf{event}(t)) &&= \textstyle\bigwedge_{\mathsf{event}(s)\in E} \neg s \doteq t\\
&tr'_{M,E}(\neg s \doteq t) &&= \neg s \doteq t\\
&tr'_{M,E}(\neg(\exists \bar{x}.\psi_1 \vee \psi_2)) &&= tr'_{M,E}(\neg\exists \bar{x}.\psi_1) \wedge tr'_{M,E}(\neg\exists \bar{x}.\psi_2)\\
&tr'_{M,E}(\neg\neg\phi) &&= tr'_{M,E}(\phi)\\
&tr'_{M,E}(\neg\exists \bar{x}.\mathsf{event}(t_1) \wedge \cdots \wedge \mathsf{event}(t_n) \wedge u_1 \doteq v_1 \wedge \cdots u_m \doteq v_m) =\\
&\quad \textstyle\bigwedge_{\mathsf{event}(s_1)\in E \ldots \mathsf{event}(s_n)\in} \neg\exists \bar{x}.\, (s_1 \doteq t_1 \wedge \cdots \wedge t_n \doteq s_n \wedge u_1 \doteq v_1 \wedge \cdots u_m \doteq v_m)
\end{aligned}$$

**Theorem 1** *Let $\mathfrak{S} = (\mathcal{S}; M; E; \phi)$ be a symbolic state and $\Psi$ a formula in the geometric fragment such that $fv(\Psi) = \emptyset$ and $var(\Psi) \cap var(\phi) = \emptyset$. For all $\mathcal{I} \models \phi$, we have $\mathcal{I}, M, E \models_{\mathfrak{S}} \neg\Psi$ iff $\mathcal{I} \models tr_{M,E}(\Psi)$. Moreover, if $\phi$ is well-formed, then so is $\phi \wedge tr_{M,E}(\Psi)$.*

### 4.2 Constraint Reduction

As mentioned before, we can transform any well-formed constraint into the form $\phi \equiv \exists \bar{x}.\phi_0 \vee \ldots \vee \phi_n$, where $\phi_i$ are base constraints, i.e., without disjunction and existential quantification (except in negative substitutions). We now discuss how to find the solutions of such well-formed base constraints. Solving intruder constraints has been studied quite extensively, e.g., in [14, 18, 5, 16], where the main application of constraints was for efficient protocol verification for a bounded number of sessions of honest agents. Here, we use constraints rather as a proof argument for the shape of attacks. Our result is of course not restricted to a bounded number of sessions as we do not rely on an exploration of reachable symbolic states (that are indeed infinite) but rather make an argument about the constraints in each of these states.

We consider *constraint reduction rules* of the form $\dfrac{\phi'}{\phi}$ (*name*), *cond* expressing that $\phi'$ entails $\phi$ (if the side condition *cond* holds). However, we will usually read the rule backwards, i.e., as: one way to solve $\phi$ is $\phi'$.

**Definition 6.** *The* satisfiability calculus for the symbolic intruder *comprises the following constraint reduction rules:*

$$\frac{eq(\sigma) \wedge \sigma(\phi)}{M \vdash t \wedge \phi} \ (\mathit{Unify}),\ s, t \notin \mathcal{V}, s \in M, \sigma \in mgu(s \doteq t) \qquad \frac{eq(\sigma) \wedge \sigma(\phi)}{s \doteq t \wedge \phi} \ (\mathit{Equation}),\ \sigma \in mgu(s \doteq t), s \notin \mathcal{V} \text{ or } s \in fv(t) \cup fv(\phi)$$

$$\frac{\phi}{M \vdash c \wedge \phi} \ (\mathit{PubConsts}),\ c \in \mathcal{C}_{pub} \qquad \frac{M \vdash t_1, \cdots, M \vdash t_n}{M \vdash f(t_1, \cdots, t_n)} \ (\mathit{Compose}),\ f \in \Sigma^n$$

$$\frac{\bigwedge_{k \in K} M \vdash k \wedge (M \vdash t \wedge \phi)^{T \gg M}}{M \vdash t \wedge \phi} \ (\mathit{Decompose}),\ s \in M, \mathtt{Ana}(s) = (K, T), T \not\subseteq M, \text{and } (\mathit{Decompose}) \text{ has not been applied with the same } M \text{ and } s \text{ before}$$

*where* $(M' \vdash t)^{T \gg M}$ *is* $M' \cup T \vdash t$ *if* $M \subseteq M'$ *and* $M' \vdash t$ *otherwise,* $eq(\sigma) = x_1 \doteq t_1 \wedge \ldots \wedge x_n \doteq t_n$ *is the constraint corresponding to a substitution* $\sigma = [x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$*, and* $mgu(s \doteq t)$ *is the standard most general unifier for the pair of terms* $t$ *and* $s$ *(in the free-algebra).*

Recall that the *mgu*, if it exists, is unique modulo renaming. We may also apply *mgu* on conjunction equations. Let us now explain the rules. (*Unify*) expresses that one way to generate a term $t$ from knowledge $M$ is to use any term $s \in M$ that can be unified with $t$, but one commits in this case to the unifier $\sigma$; this is done by applying $\sigma$ to the rest of the constraint and recording its equations. (*Unify*) cannot be applied when $s$ or $t$ are variables; intuitively: when $t$ is a variable, the solution is an arbitrary term, so we consider this a solved state (until elsewhere a substitution is required that substitutes $t$); when $s$ is variable, then it is a subterm of a message that the intruder created earlier. If the earlier constraint is already solved (i.e., a variable) then $s$ is something the intruder could generate from an earlier knowledge and thus redundant.

(*Equation*), which similarly allows us to solve an equation, can be applied if $s$ or $t$ are variables, provided the conditions are satisfied, but later we will have to prevent vacuous application of this rule to its previous result, i.e., the equations $eq(\sigma)$. (*PubConsts*) says that the intruder can generate all public constants.

(*Compose*) expresses that one way to generate a composed term $f(t_1, \ldots, t_n)$ is to generate the subterms $t_1, \ldots, t_n$ (because then $f$ can be applied to them). (*Decompose*) expresses that we can attempt decryption of any term in the intruder knowledge according to the `Ana` function. Recall that Table 1 provides examples of `Ana`, and note that for variables or constants Table 1 will yield $(\emptyset, \emptyset)$, i.e., there is nothing to analyze. However, if there is a set $T$ of messages that can potentially be obtained if we can derive the keys $K$, and $T$ is not yet a subset of the knowledge $M$ anyway, then one way to proceed is to add $M \vdash k$ for each

$k \in K$ to the constraint store, i.e., committing to finding the keys, and under this assumption we may add $T$ to $M$ and in fact to any knowledge $M'$ that is a superset of $M$. Also for this rule we must prevent vacuous repeated application, such as applying analysis directly to the newly generated $M \vdash k$ constraints.

The reduction of constraints deals with conjuncts of the form $M \vdash t$ and $s \doteq t$. However, we also have to handle negative substitutions, i.e., conjuncts of the form $\neg\exists \bar{x}.\phi_\sigma$. We show we can easily check them for satisfiability.

**Definition 7.** *A constraint $\phi$ is* simple*, written simple$(\phi)$, iff $\phi = \phi_1 \wedge \ldots \wedge \phi_n$ such that for each $\phi_i$ $(1 \leq i \leq n)$:*

- *if $\phi_i = M \vdash t$, then $t \in \mathcal{V}$;*
- *if $\phi_i = s \doteq t$, then $s \in \mathcal{V}$ and $s$ does not appear elsewhere in $\phi$;*
- *if $\phi_i = \neg\exists \bar{x}.\phi_\sigma$, then $mgu(\theta(\phi_\sigma)) = \emptyset$ for $\theta = [\bar{y} \mapsto \bar{c}]$ where $\bar{y}$ are the free variables of $\phi_i$ and $\bar{c}$ fresh constants that do not appear in $\phi$.*

**Theorem 2** *If simple$(\phi)$, then $\phi$ is satisfiable.*

**Theorem 3 (Adaption of [18, 16])** *The satisfiability calculus for the symbolic intruder is sound, complete, and terminating on well-formed constraints.*

## 5 Typed Model

In our typed model, the *set of all possible types for terms* is denoted by $\mathcal{T}_{\Sigma \cup \mathfrak{T}_a}$, where $\mathfrak{T}_a$ is a finite set of *atomic types*, e.g., $\mathfrak{T}_a = \{Number, Agent, PublicKey, PrivateKey, SymmetricKey\}$. We call all other types *composed types*. Each atomic term (each element of $\mathcal{V} \cup \mathcal{C}$) is given a type; constants are given an *atomic type* and variables are given either an atomic or a composed type (any element of $\mathcal{T}_{\Sigma \cup \mathfrak{T}_a}$). We write $t : \tau$ to denote that a term $t$ has the type $\tau$. Based on the type information of atomic terms, we define the *typing function* $\Gamma$ as follows:

**Definition 8.** *Given $\Gamma(\cdot) : \mathcal{V} \to \mathcal{T}_{\Sigma \cup \mathfrak{T}_a}$ for variables and $\Gamma(\cdot) : \mathcal{C} \to \mathfrak{T}_a$ for constants, we extend $\Gamma$ to map all terms to a type, i.e., $\Gamma(\cdot) : \mathcal{T}_{\Sigma \cup \mathcal{C}}(\mathcal{V}) \to \mathcal{T}_{\Sigma \cup \mathfrak{T}_a}$, as follows: $\Gamma(t) = f(\Gamma(t_1), \cdots, \Gamma(t_n))$ if $t = f(t_1, \cdots, t_n)$ and $f \in \Sigma^n$. We say that a substitution $\sigma$ is* well-typed *iff $\Gamma(x) = \Gamma(\sigma(x))$ for all $x \in dom(\sigma)$.*

For example, if $\Gamma(k) = PrivateKey$ and $\Gamma(x) = Number$ then $\Gamma(\mathsf{crypt}(\mathsf{pub}(k), x)) = \mathsf{crypt}(\mathsf{pub}(PrivateKey), Number)$.

As we require that all constants be typed, we further partition $\mathcal{C}$ into disjoint countable subsets according to different types in $\mathfrak{T}_a$. This models the intruder's ability to access infinite reservoirs of public fresh constants. For example, for protocols $P_1, P_2$ and $\mathfrak{T}_a = \{\beta_1, \ldots, \beta_n\}$, we have the disjoint subsets $\mathcal{C}_{pub}^{\beta_i}$, $\mathcal{C}_{priv}^{\beta_i}$, $\mathcal{C}_{P_1}^{\beta_i}$ and $\mathcal{C}_{P_2}^{\beta_i}$, where $i \in \{1, \ldots, n\}$ and, e.g., $\mathcal{C}_{pub}^{\beta_i}$ contains all $\mathcal{C}_{pub}$ elements of type $\beta_i$. $\mathcal{C}_{P_1}^{\beta_i}$ and $\mathcal{C}_{P_2}^{\beta_i}$ are short-term constants, whereas $\mathcal{C}_{pub}^{\beta_i}$ and $\mathcal{C}_{priv}^{\beta_i}$ are long-term, and we consider it an attack if the intruder learns any of $\mathcal{C}_{priv}^{\beta_i}$.

By an easy induction on the structure of terms, we have:

**Lemma 1** *If a substitution $\sigma$ is well-typed, then $\Gamma(t) = \Gamma(\sigma(t))$ for all terms $t \in \mathcal{T}_{\Sigma \cup \mathcal{C}}(\mathcal{V})$.*

According to this typed model, $\mathcal{I}$ is a *well-typed interpretation* iff $\Gamma(x) = \Gamma(\mathcal{I}(x))$ for all $x \in \mathcal{V}$. Moreover, we require for the typed model that $\Gamma(s) = \Gamma(t)$ for each $s \doteq t$. This is a restriction only on the strands of the honest agents (as they are supposed to act honestly), not on the intruder: he can send ill-typed messages freely. We later show that sending ill-typed messages does not help the intruder in introducing new attacks in protocols that satisfy certain conditions.

### 5.1 Message Patterns

In order to prevent the intruder from using messages of a protocol to attack a second protocol, we need to guarantee the disjointness of the messages between both protocols. Thus, we use formats to wrap raw data, as discussed in § 2.2. In particular, all submessages of all operators (except formats and public key operator) must be "wrapped" with a format, e.g., $\mathsf{scrypt}(k, \mathsf{f}_a(Na))$ and $\mathsf{scrypt}(k, \mathsf{f}_b(Nb))$ should be used instead of $\mathsf{scrypt}(k, Na)$ and $\mathsf{scrypt}(k_1, Nb)$.

We define the set of protocol message patterns, where we need to ensure that each pair of terms has disjoint variables: we thus define a well-typed $\alpha$-renaming $\alpha(t)$ that replaces the variables in $t$ with completely new variable names.

**Definition 9.** *The* message pattern *of a message $t$ is $MP(t) = \{\alpha(t)\}$. We extend $MP$ to strands, goals and protocols as follows. The* set $MP(S)$ of message patterns of a strand $S$ *and the* set $MP(\Psi)$ of message patterns of a goal $\Psi$ *are defined as follows:*

$$
\begin{array}{llll}
MP(\mathsf{send}(t).S) & = MP(t) \cup MP(S) & MP(\forall x.\psi \Rightarrow \psi_0) & = MP(\psi) \cup MP(\psi_0) \\
MP(\mathsf{event}(t).S) & = MP(t) \cup MP(S) & MP(\mathsf{ik}(t)) & = MP(t) \\
MP(\mathsf{receive}(t).S) & = MP(t) \cup MP(S) & MP(\mathsf{event}(t)) & = MP(t) \\
MP(s \doteq t.S) & = MP(\sigma(S)),\ \textit{for}\ \sigma \in mgu(s \doteq t) & MP(\psi_1 \vee \psi_2) & = MP(\psi_1) \cup MP(\psi_2) \\
MP(s \doteq t.S) & = \emptyset\ \textit{if}\ mgu(s \doteq t) = \emptyset & MP(\psi_1 \wedge \psi_2) & = MP(\psi_1) \cup MP(\psi_2) \\
MP((\neg\exists\bar{x}.\phi_\sigma).S) & = MP(\phi_\sigma) \cup MP(S) & MP(s \doteq t) & = MP(s) \cup MP(t) \\
MP(0) & = \emptyset & MP(\neg\phi) & = MP(\phi)
\end{array}
$$

*The* set of message patterns of a protocol $P = (\{S_1, \cdots, S_m\}; \{\Psi_0, \cdots, \Psi_n\})$ *is* $MP(P) = \bigcup_{i=1}^{m} MP(S_i) \cup \bigcup_{i=1}^{n} MP(\Psi_i)$*, and the* set of sub-message patterns of a protocol $P$ *is* $SMP(P) = \{\alpha(s) \mid t \in MP(P) \wedge s \sqsubseteq t \wedge \neg atomic(s)\} \setminus \{u \mid u = pub(v)$ *for some term* $v\}$*. SMP applies to messages, strands, goals as expected.*

*Example 2.* If $S = \mathsf{receive}(\mathsf{scrypt}(k, (\mathsf{f}_1(x, y)))).\mathsf{send}(\mathsf{scrypt}(k, y))$, then $SMP(S) = \{\mathsf{scrypt}(k, \mathsf{f}_1(x_1, y_1)), \mathsf{scrypt}(k, y_2), \mathsf{f}_1(x_3, y_3)\}$. □

**Definition 10.** *A protocol $P = (\mathcal{S}_0, G)$ is* type-flaw-resistant *iff the following conditions hold:*

- *$MP(P)$ and $\mathcal{V}$ are disjoint, i.e., $MP(P) \cap \mathcal{V} = \emptyset$ (which ensures that none of the messages of $P$ is sent as raw data).*

- *If two non-atomic sub-terms are unifiable, then they have the same type, i.e., for all $t_1, t_2 \in SMP(P)$, if $\sigma(t_1) = \sigma(t_2)$ for some $\sigma$, then $\Gamma(t_1) = \Gamma(t_2)$.*
- *For any equation $s \doteq t$ that occurs in strands or goals of $P$ (also under a negation), $\Gamma(s) = \Gamma(t)$.*
- *For every variable $x$ that occurs in equations or events of $G$, $\Gamma(x) \in \mathfrak{T}_a$.*
- *For every variable $x$ that occurs in inequalities or events of strands, $\Gamma(x) \in \mathfrak{T}_a$.*

Intuitively, the second condition means that we cannot unify two terms unless their types match. Note that this match is a restriction on honest agents only, the intruder is still able to send ill-typed messages.

*Example 3.* Example 2 included a potential type-flaw vulnerability, since $\mathsf{scrypt}(k, \mathsf{f}_1(x_1, y_1))$ and $\mathsf{scrypt}(k, y_2)$ have the unifier $[y_2 \mapsto \mathsf{f}_1(x_1, y_1)]$. Here $y_1$ and $y_2$ must have the same type since they have been obtained by a well-typed variable renaming in the construction of $SMP$. Thus, the two messages have different types. The problem is that the second message encrypts raw data without any information on who it is meant for and it may thus be mistaken for the first message. Let us thus change the second message to $\mathsf{scrypt}(k, \mathsf{f}_2(y_2))$. Then $SMP$ also includes $\mathsf{f}_2(y_4)$ for a further variable $y_4$, and now no two different elements of $SMP$ have a unifier. $\mathsf{f}_2$ is not necessarily inserting a tag: if the type of $y$ in the implementation is a fixed-length type, this is already sufficient for distinction. □

**Theorem 4** *If a type-flaw-resistant protocol $P$ has an attack, then $P$ has a well-typed attack.*

Note that this theorem does not exclude that type-flaw attacks are possible, but rather says that for every type-flaw attack there is also a (similar) well-typed attack, so it is safe to verify the protocol only in the typed model.

## 6 Parallel Composition

In this section, we consider the parallel composition of protocols, which we often abbreviate simply to "composition". We define the set of operational strands for the composition of a pair of protocols as the union of the sets of the operational strands of the two protocols; this allows all possible transitions in the composition. The goals for the composition are also the union of the goals of the pair, since any attack on any of them is an attack on the whole composition (i.e., the composition must achieve the goals of the pair).

**Definition 11.** *The* parallel composition $P_1 \parallel P_2$ *of* $P_1 = (S_0^{P_1}; \Psi_0^{P_1})$ *and* $P_2 = (S_0^{P_2}; \Psi_0^{P_2})$ *is* $P_1 \parallel P_2 = (S_0^{P_1} \cup S_0^{P_2}; \Psi_0^{P_1} \cup \Psi_0^{P_2})$.

Our parallel composition result relies on the following key idea. Similar to the typing result, we look at the constraints produced by an attack trace against $P_1 \parallel P_2$, violating a goal of $P_1$, and show that we can obtain an attack against $P_1$ alone, or a violation of a long-term secret by $P_2$. Again, the core of this proof is the observation that the unification steps of the symbolic intruder never produce an "ill-typed" substitution in the sense that a $P_1$-variable is never instantiated with a $P_2$ message and vice versa. For that to work, we have a similar condition as before, namely that the non-atomic subterms of the two protocols (the SMPs) are disjoint, i.e., each non-atomic message uniquely says to which protocol it belongs. This is more liberal than the requirements in previous parallel compositionality results in that we do not require a particular tagging scheme: any way to make the protocol messages distinguishable is allowed. Further, we carefully set up the use of constants in the protocol as explained at the beginning of § 5, namely that all constants used in the protocol are: long-term public values that the intruder initially knows; long-term secret values that, if the intruder obtains them, count as a secrecy violation in *both* protocols; or short-term values of $P_1$ or of $P_2$.

The only limitation of our model is that long-term secrets cannot be "declassified": we require that all constants of type private key are either part of the long-term secrets or long-term public constants. Moreover, the intruder can obtain all public keys, i.e., $\mathsf{pub}(c)$ for every $c$ of type private key. This does not prevent honest agents from creating fresh key-pairs (the private key shall be chosen from the long-term constants as well) but it dictates that each private key is either a perpetual secret (it is an attack if the intruder obtains it) or it is public right from the start (as all public keys are). This only excludes protocols in which a private key is a secret at first and later revealed to the intruder, or where some public keys are initially kept secret.

**Definition 12.** *Two protocols $P_1$ and $P_2$ are* parallel-composable *iff the following conditions hold:*

*(1) $P_1$ and $P_2$ are SMP-*disjoint*, i.e., for every $s \in SMP(P_1)$ and $t \in SMP(P_2)$, either $s$ and $t$ have no unifier ($mgu(s \doteq t) = \emptyset$) or $s = \mathsf{pub}(s_0)$ and $t = \mathsf{pub}(t_0)$ for some $s_0, t_0$ of type private key.*
*(2) All constants of type private key that occur in $MP(P_1) \cup MP(P_2)$ are part of the long-term constants in $\mathcal{C}_{pub} \cup \mathcal{C}_{priv}$.*
*(3) All constants that occur in $MP(P_i)$ are in $\mathcal{C}_{pub} \cup \mathcal{C}_{priv} \cup \mathcal{C}_{P_i}$, i.e., are either long term or belong to the short-term constants of the respective protocol.*
*(4) For every $c \in \mathcal{C}_{P_i}^{PrivateKey}$, $P_i$ also contains the strand $\mathsf{send}(\mathsf{pub}(c)).0$.*
*(5) For each secret constant $c \in \mathcal{C}_{priv}^{\beta_i}$, for each type $\beta_i$, each $P_i$ contains the strands $\mathsf{event}(lts_{\beta_i,P_i}(c)).0$ and the goal $\forall x : \beta_i .\, \mathsf{ik}(x) \implies \neg lts_{\beta,P_i}(x)$.*
*(6) Both $P_1$ and $P_2$ are type-flaw resistant.*

Some remarks on the conditions: (1) is the core of the compositionality result, as it helps to avoid confusion between messages of the two protocols; (2) ensures that every private key is either initially known to the intruder or is part of the long-term secrets (and thus prevents "declassification" of private keys as we discussed above). (3) means that the two protocols will draw from disjoint sets of constants for their short-term values. (4) ensures that public keys are known to the intruder. Note that typically the goals on long-term secrets, like private keys and shared symmetric keys, are very easy to prove as the are normally not transmitted. The fact that we do not put all public keys into the knowledge of the intruder in the initial state is because the intruder knowledge must be a finite set of terms for the constraint reduction to work. Putting it into strands means they are available at any time, but the intruder knowledge in every reachable state (and thus constraint) is finite. Similarly, for the goals on long-term secrets: the set of events in every reachable state is still finite, but for every leaked secret, we can in one transition reach the corresponding predicate that triggers the secrecy violation goal. (5) ensures that when either protocol $P_i$ leaks any constant of $\mathcal{C}^{\beta_i}_{priv}$, it is a violation of its secrecy goals. (6) ensures that for both protocols, we cannot unify terms unless their types match.

**Theorem 5** *If two protocols $P_1$ and $P_2$ are parallel-composable and both $P_1$ and $P_2$ are secure in isolation in the typed model, then $P_1 \parallel P_2$ is secure (also in the untyped model).*

We can then apply this theorem successively to any number of protocols that satisfy the conditions, in order to prove that they are all parallel composable.

This compositionality result entails an interesting observation about parallel composition with insecure protocols: unless one of the protocols leaks a long-term secret, the intruder never needs to use one protocol to attack another protocol. This means actually: even if a protocol is flawed, it does not endanger the security of the other protocols as long as it at least manages not to leak the long-term secrets. For instance, the Needham-Schroeder Public Key protocol has a well-known attack, but the intruder can never obtain the private keys of any honest agent. Thus, another protocol relying on the same public-key infrastructure is completely unaffected. This is a crucial point because it permits us to even allow for security statements in presence of flawed protocols:

**Corollary 1.** *Consider two protocols $P_1$ and $P_2$ that are parallel-composable (and thus satisfy all the conditions in Definition 12). If $P_1$ is secure in isolation and $P_2$, even though it may have an attack in isolation, does not leak a long-term secret, then all goals of $P_1$ hold also in $P_1 \parallel P_2$.*

## 7 Tool support

We have developed the *Automated Protocol Composition Checker* APCC[6], a tool that verifies the two main syntactic conditions of our results: it checks both if a given protocol is type-flaw-resistant and if the protocols in a given set are pairwise parallel-composable. In our preliminary experiments, we considered a suite that includes widely used protocols like TLS, Kerberos (PKINIT and Basic) and protocols defined by the ISO/IEC 9798 standard, along with well-known academic protocols (variants of Needham-Schroeder-Lowe, Denning-Sacco, etc.). Although we worked with abstract and simplified models, we were able to verify that TLS and Kerberos are parallel-composable. In contrast, since some protocols of the ISO/IEC 9798 standard share common formats, they are not *SMP*-disjoint.

[6] Available at `http://www2.compute.dtu.dk/~samo/APCC.zip`.

Another result is that many academic protocols are not pairwise parallel-composable. This was largely expected because they do not have a standardized implementation, and thus the format of messages at the wire level is not part of the specification. In fact, in these protocols there are several terms that may be confused with terms of other protocols, whereas a concrete implementation may avoid this by choosing carefully disjoint messages formats that can prevent the unification. Hence, our tool APCC can also support developers in the integration of new protocols (or new implementations of them) in an existing system.

## 8 Conclusions and Related Work

This paper unifies research on the soundness of typed models [11, 4, 16, 2] and on parallel protocol composition [10, 8, 6, 7, 1] by using a proof technique that has been employed in both areas: attack reduction based on a symbolic constraint systems. For typing, the idea is that the constraint solving never needs to apply ill-typed substitutions if the protocol satisfies some sufficient conditions; hence, for every attack there exists a well-typed variant and it is thus without loss of generality to restrict the model to well-typed execution. For the parallel composition of $P_1$ and $P_2$ that again satisfy some sufficient conditions, the constraint solving never needs to use a message that the intruder learned from $P_1$ to construct a message of $P_2$; thus, the attack will work in $P_1$ alone or in $P_2$ alone, and from verifying them in isolation, we can conclude that their composition is secure.

We also generalize over previous results. First, we are not limited to pure secrecy goals, but consider the entire geometric fragment proposed by Guttman [9]. One could characterize this fragment as allowing a "controlled" amount of negation in the goal specifications. In fact, we believe this is the most expressive language that will work with the given constraint solving argument.

Another generalization over many previous works concerns the handling of keys: long-term keys can be shared amongst protocols, keys can be freshly generated and transmitted (including public keys), and keys may be composed.

While many previous works require protocols to implement particular tagging schemes (to identify either the intended type of message or which protocol it belongs to), we have a more liberal requirement: non-atomic subterms should not have a unifier unless they have the same type and belong to the same protocol.

This is especially critical for applying the results to real-world protocols like TLS that often do not follow the tagging schemes of a particular approach. In fact, TLS (in its basic form without extensions and optional sides) satisfies our sufficient conditions for typing. While our sufficient conditions are basically syntactic properties of message patterns, checking them is tedious for large protocols and sets of protocols. For this reason, and to make the work useable also to security protocol designers without formal background, we have been developing the APCC tool along with an ever-growing library of securely composable protocols.

Our work considered so far protocols only in the initial term algebra without any algebraic properties. There are some promising results in this direction [12, 7, 15] that we would like to combine with our approach. The same holds for other types of protocol composition, e.g., the sequential composition considered in [7], where one protocol establishes a key that is used by another protocol as input.

## References


1. S. Andova, C. J. F. Cremers, K. Gjøsteen, S. Mauw, S. F. Mjølsnes, and S. Radomirovic. A framework for compositional verification of security protocols. *Inform. Comput.*, 206(2-4):425–459, 2008.
2. M. Arapinis and M. Duflot. Bounding messages for free in security protocols - extension to various security properties. *Inform. Comput.*, 239:182–215, 2014.
3. A. Armando and L. Compagna. SATMC: A SAT-based Model Checker for Security Protocols. *Logics in Artificial Intelligence*, pp. 730–733, 2004.
4. B. Blanchet and A. Podelski. Verification of cryptographic protocols: tagging enforces termination. *Theor. Comput. Sci.*, 333(1-2):67–90, 2005.
5. H. Comon-Lundh, S. Delaune, and J. K. Millen. Constraint solving techniques and enriching the model with equational theories. In *Formal Models and Techniques for Analyzing Security Protocols*, pp. 35–61. IOS Press, 2011.
6. V. Cortier and S. Delaune. Safely composing security protocols. *Form. Method. Syst. Des.*, 34:1–36, 2009.
7. Ştefan Ciobâcă and V. Cortier. Protocol composition for arbitrary primitives. In *CSF*, pp. 322–336. IEEE, 2010.
8. J. D. Guttman. Cryptographic Protocol Composition via the Authentication Tests. In *FOSSACS'09*, pp. 303–317. Springer, 2009.
9. J. D. Guttman. Establishing and preserving protocol security goals. *Journal of Computer Security*, 22(2):203–267, 2014.
10. J. D. Guttman and F. J. Thayer. Protocol independence through disjoint encryption. In *CSFW*, pp. 24–34. IEEE, 2000.
11. J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. *Journal of Computer Security*, 11(2):217–244, 2003.
12. R. Küsters and T. Truderung. Using ProVerif to Analyze Protocols with Diffie-Hellman Exponentiation. In *CSF*, pp. 157–171. IEEE, 2009.
13. G. Lowe. A hierarchy of authentication specifications. In *CSFW*, pp. 31–44, 1997.
14. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *CCS*, pp. 166–175. ACM, 2001.
15. S. Mödersheim. Diffie-Hellman without Difficulty. In *FAST*, pp. 214–229, 2011.
16. S. Mödersheim. Deciding Security for a Fragment of ASLan. In *ESORICS*, pp. 127–144. Springer, 2012.
17. S. Mödersheim and G. Katsoris. A sound abstraction of the parsing problem. In *CSF*, pp. 259–273. IEEE, 2014.
18. M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions and composed keys is NP-complete. *Theor. Comput. Sci.*, 299, 2003.
19. F. J. Thayer, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1):191–230, 1999.

## A Appendix: proofs of the technical results

**Theorem 1** *Let $\mathfrak{S} = (\mathcal{S}; M; E; \phi)$ be a symbolic state and $\Psi$ a formula in the geometric fragment such that $fv(\Psi) = \emptyset$ and $var(\Psi) \cap var(\phi) = \emptyset$. For all $\mathcal{I} \models \phi$, we have $\mathcal{I}, M, E \models_{\mathfrak{S}} \neg\Psi$ iff $\mathcal{I} \models tr_{M,E}(\Psi)$. Moreover, if $\phi$ is well-formed, then so is $\phi \wedge tr_{M,E}(\Psi)$.*

*Proof.* We first prove, by induction, a corresponding property for the function $tr'$ that is called by the $tr$ function. For that assume we have $\mathcal{I}$, $M$, $E$, $\phi$, and $\psi$ such that $\mathcal{I} \models \phi$, $var(\phi) \cap var(\psi) = \emptyset$, $fv(\phi) \subseteq fv(tr'_{M,E}(\psi)) \subseteq fv(\phi) \cup fv(\psi)$, $fv(\phi) = var(M) \cup var(E)$. Also we have that $E = \{\mathsf{event}(s_1), \cdots, \mathsf{event}(s_n)\}$ since $E$ is a finite set. We prove $\mathcal{I}, M, E \models_{\mathfrak{S}} \psi$ iff $\mathcal{I} \models tr'_{M,E}(\psi)$ by induction on the structure of $tr'_{M,E}(\cdot)$:

- $\mathcal{I}, M, E \models_{\mathfrak{S}} \mathsf{ik}(t)$ iff $\mathcal{I}(M) \vdash \mathcal{I}(t)$ iff $\mathcal{I} \models M \vdash t = tr'_{M,E}(\mathsf{ik}(t))$.
- $\mathcal{I}, M, E \models_{\mathfrak{S}} \mathsf{event}(t)$ iff $\mathcal{I}(\mathsf{event}(t)) \in \mathcal{I}(E)$ iff $I(t) \in \{\mathcal{I}(s_1), \cdots, \mathcal{I}(s_n))\}$ iff $\mathcal{I}(t) = \mathcal{I}(s_1) \vee \cdots \vee \mathcal{I}(t) = \mathcal{I}(s_n)$ iff $\mathcal{I} \models t \doteq s_1 \vee \cdots \vee t \doteq s_n$ iff $\mathcal{I} \models \bigvee_{\mathsf{event}(s) \in E} s \doteq t = tr'_{M,E}(\mathsf{event}(t))$.
- $\mathcal{I}, M, E \models_{\mathfrak{S}} \psi_1 \vee \psi_2$ iff $\mathcal{I}, M, E \models_{\mathfrak{S}} \psi_1$ or $\mathcal{I}, M, E \models_{\mathfrak{S}} \psi_2$ iff $\mathcal{I} \models \psi_1$ or $\mathcal{I} \models \psi_2$ by induction iff $\mathcal{I} \models tr'_{M,E}(\psi_1) \vee tr'_{M,E}(\psi_2) = tr'_{M,E}(\psi_1 \vee \psi_2)$.
- The other cases follow similarly.

Based on this, we prove $\mathcal{I}, M, E \models \neg\Psi$ iff $\mathcal{I} \models tr_{M,E}(\Psi)$. Let $\Psi = \forall \bar{x}.\psi \implies \psi_0$. Then, $\mathcal{I} \models tr_{M,E}(\Psi) = \exists \bar{x}.\, tr'_{M,E}(\psi) \wedge tr'_{M,E}(\neg\psi_0)$ iff

- exist $\bar{t}$ such that $\mathcal{I}[\bar{x} \mapsto \bar{t}] \models tr'_{M,E}(\psi)$ and $\mathcal{I}[\bar{x} \mapsto \bar{t}] \models tr'_{M,E}(\neg\psi_0)$ iff
- exist $\bar{t}$ such that $\mathcal{I}[\bar{x} \mapsto \bar{t}], M, E \models_{\mathfrak{S}} \psi$ and $\mathcal{I}[\bar{x} \mapsto \bar{t}], M, E \models_{\mathfrak{S}} \neg\psi_0$ iff
- exist $\bar{t}$ such that $\mathcal{I}[\bar{x} \mapsto \bar{t}], M, E \models_{\mathfrak{S}} \psi \wedge \neg\psi_0$ iff
- $\mathcal{I}, M, E \models_{\mathfrak{S}} \exists \bar{x}.\psi \wedge \neg\psi_0$ iff
- $\mathcal{I}, M, E \models_{\mathfrak{S}} \neg\forall \bar{x}.\psi \implies \psi_0 = \neg\Psi$.

The well-formedness follows from the fact that in each state, the knowledge $M$ is a superset of every $M'$ that occur in a deduction constraint $M' \vdash t$ in $\phi$. Further, $M$ can only contain variables that occur in some $t$ for which $M' \vdash t$ occurs in $\phi$. Thus, $tr_{M,E}(\Psi) \wedge \phi$ is well-formed, if $\phi$ is. ☐

**Theorem 2** *If $simple(\phi)$, then $\phi$ is satisfiable.*

*Proof.* From $simple(\phi)$, by the definition of *simple* (Definition 7), it follows that $\phi$ is a conjunction of intruder deduction constraints of the form $M \vdash x$ with $x \in \mathcal{V}$, equations $x \doteq t$ where $x \in \mathcal{V}$ and where $x$ does not occur elsewhere in $\phi$, and inequalities. Let $\bar{y}$ be all variables that occur freely in intruder deduction constraints and inequalities, and let $\theta = [\bar{y} \mapsto \bar{c}]$ for new constants $\bar{c} \in \mathcal{C}_{pub}$ (that do not occur in $\phi$ and are pairwise different). We show that $\theta(\phi)$ is satisfiable.

All intruder deduction constraints are satisfiable since the constants are in $\mathcal{C}_{pub}$ and the intruder can access those constants by the rule (*Public*) as in Definition 1.

The equations are obviously satisfiable: all equations in $\phi$ have the form $v_i \doteq u_i$, with variables $\bar{v}$ that do not occur elsewhere in $\phi$, which implies that

$dom(\theta) \cap \bar{v} = \emptyset$, and thus that $\theta(v_i \doteq u_i) = v_i \doteq \theta(u_i)$. All these equations are therefore satisfiable by instantiating every $v_i \in \bar{v}$ with the term $u_i$.

It remains to show that the inequalities are satisfiable under $\theta$. Let $\phi_0 = \neg\exists\bar{x}.\,\phi_\sigma$ with $\phi_\sigma = \bigwedge s_i \doteq t_i$ be any inequality. $\theta(\phi_0)$ is closed, i.e., $fv(\theta(\phi_0)) = \emptyset$. This implies that $fv(\phi_\sigma) = \{\bar{x}\}$, and since $\phi$ is simple, we have $mgu(\theta(\phi_\sigma)) = \emptyset$. Then, $\phi_\sigma$ is not satisfiable, i.e., there do not exist $\bar{x}$ such that $\phi_\sigma$ holds. Thus, $\phi_0$ holds. □

The completeness of the symbolic intruder constraint reduction is similar to existing results on symbolic intruder constraints; what is particular is our generalization to constraints with quantified inequalities. To that end, we first show:

**Lemma 2** *Let $\phi = \neg\exists\bar{x}.\phi_\sigma$ where $\phi_\sigma = \bigwedge s_i \doteq t_i$, and let $\theta = [\bar{y} \mapsto \bar{c}]$ where $\bar{y} = fv(\phi)$ and $\bar{c}$ are fresh public constants that do not occur in $\phi$. Then $\phi$ is satisfiable iff $\theta(\phi)$ is satisfiable. Moreover, $\phi$ is satisfiable iff $mgu(\theta(\phi_\sigma)) = \emptyset$.*

*Proof.* If $\theta(\phi)$ is unsatisfiable, then also $\phi$ is unsatisfiable. For the other direction, we show that the following two formulas are a contradiction:

$$\exists\bar{y}.\forall\bar{x}.\bigvee_{i=1}^{n} s_i \neq t_i \tag{1}$$

$$\exists\bar{x}.\bigwedge_{i=1}^{n} \theta(s_i) = \theta(t_i) \tag{2}$$

By (2), we can find a substitution $\xi = [\bar{x} \mapsto \bar{u}]$ where $\bar{u}$ are ground terms such that $\bigwedge_{i=1}^{n} \xi(\theta(s_i)) = \xi(\theta(t_i))$. Since $\theta$ and $\xi$ are substitutions with disjoint domain and grounding, we have $\theta(\xi(\cdot)) = \xi(\theta(\cdot))$, and thus we obtain

$$\bigwedge_{i=1}^{n} \theta(\xi(s_i)) = \theta(\xi(t_i)) \tag{3}$$

By (1), choosing a particular value for the $\bar{x}$, we obtain:

$$\exists\bar{y}.\bigvee_{i=1}^{n} \xi(s_i) \neq \xi(t_i) \tag{4}$$

Then we can find an $i \in \{1, ..., n\}$ such that $\exists\bar{y}.\,\xi(s_i) \neq \xi(t_i)$. Thus, taking $s := \xi(s_i)$ and $t := \xi(t_i)$, we have:

$$\exists\bar{y}.\,s \neq t \tag{5}$$

$$\theta(s) = \theta(t) \tag{6}$$

To show that (5) and (6) yield a contradiction, we consider all possible cases of $s$ and $t$:

– If $s$ and $t$ are atomic, then, since $\theta$ replaces all of variables $\bar{y}$ with fresh constants, $\theta(s) = \theta(t)$ implies $s = t$, contradicting (5).

– If $s$ is atomic and $t$ is not, then, since $\theta(s)$ is a constant, $\theta(s) \neq \theta(t)$, contradicting (6).
– If both $s$ and $t$ are not atomic, then $s = f(s_1, \ldots, s_n)$ and $t = f(t_1, \ldots, t_n)$ (otherwise $\theta(s) = \theta(t)$ cannot hold). Thus, we can reduce this case to one pair $s_i$ and $t_i$ of corresponding subterms.

Now, since $\theta(\phi)$ is closed, i.e., $fv(\phi_\sigma) = \bar{x}$, we can decide the satisfiability of $\phi$ with the *mgu*-algorithm. □

**Theorem 3 (Adaption of [18, 16])** *The satisfiability calculus for the symbolic intruder is sound, complete, and terminating on well-formed constraints.*

*Proof.* Let us write $\phi \rightsquigarrow \phi'$ if $\dfrac{\phi'}{\phi}$ is an instance of a reduction rule, i.e., representing one solution step. Soundness is easy since for each rule $\dfrac{\phi'}{\phi}$, from a satisfying interpretation of an instance $\sigma(\phi')$ of $\phi'$, we can derive an interpretation that satisfies $\sigma(\phi)$.

The hard part is completeness, i.e., when $\mathcal{I} \models \phi$, then either $\phi$ is already simple or we can apply some rule, obtaining $\phi \rightsquigarrow \phi'$ for some $\phi'$ with $\mathcal{I} \models \phi'$. Thus, we show that every solution $\mathcal{I}$ of a constraint is preserved by at least one applicable reduction rule until we obtain a simple constraint (that we already know is satisfiable by Theorem 2). Consider a satisfiable non-simple constraint $\phi$, and a satisfying interpretation $\mathcal{I}$. Since $\mathcal{I}$ satisfies $\phi$, for every intruder deduction $M \vdash t$ in $\phi$, there exists a proof $\mathcal{I}(M) \vdash \mathcal{I}(t)$ using the intruder deduction rules of Definition 1. This proof has a tree shape with $\mathcal{I}(M) \vdash \mathcal{I}(t)$ at the root and axioms as leaves for members of $\mathcal{I}(M)$. We label each $M \vdash t$ with such a proof for $\mathcal{I}(M) \vdash \mathcal{I}(t)$.

We now proceed from the first (in the order induced by the well-formedness of $\phi$) intruder constraint $M \vdash t$ where $t \notin \mathcal{V}$ (i.e., not yet simple) and show: depending on the form of the derivation tree, we can pick a rule so that we can label all new deduction constraints in the resulting constraint $\phi'$ again with matching proof trees, i.e., so that they support still the solution. In particular, we will apply the (*Unify*) rule only with substitutions of which $\mathcal{I}$ is an instance.

If $\phi$ contains a non-simple equation, i.e., $s \doteq t$ where neither $s$ nor $t$ is a variable that does not occur in $\phi$, then we can apply the (*Equation*) rule to simplify it, because $\phi$ is satisfiable under $\mathcal{I}$. Thus, $\mathcal{I}(s) = \mathcal{I}(t)$ and so there is a $\sigma \in mgu(s \doteq t)$, with $\mathcal{I}(x) = \mathcal{I}(\sigma(x))$ for all $x \in \mathcal{V}$. Therefore, the resulting constraint (replacing $s \doteq t$ by $eq(\sigma)$ and applying $\sigma$ to the rest of the constraint) still has $\mathcal{I}$ as a model.

If all equations are simple, then for $\phi$ to be non-simple, there must be at least one conjunct $M_i \vdash t_i$ where $t_i \notin \mathcal{V}$. Consider the smallest such $i$ (in the order of the well-formedness of $\phi$, thus $fv(M_i) \subseteq \{t_1, \ldots, t_{i-1}\} \subseteq \mathcal{V}$). Moreover, consider the ground derivation of $\mathcal{I}(M_i) \vdash \mathcal{I}(t_i)$, which exists because $\phi$ is satisfiable. We distinguish the different cases at the root of this proof tree:

– If it is a leaf, then $\mathcal{I}(t_i) \in \mathcal{I}(M_i)$, thus $t_i$ has a unifier with some term $s \in M_i$. Now, $t_i$ cannot be a variable because otherwise this conjunct would be already simple). If $s$ is a variable, then $s = t_j$ for some $j < i$, and we can thus proceed by following the proof tree of $t_j$ instead. If neither $t_i$ nor $s$ are variables, then the (*Unify*) rule is applicable, and again the unifier $\sigma$ supports $\mathcal{I}$, and so does the resulting constraint.

– If it is an application of the (*Public*) rule, then $t_i \in \mathcal{C}_{pub}$ and so the public constant rule of the constraint reduction is applicable.
– If it is an application of the (*Compose*) rule, then so is the corresponding rule of the constraint reduction, producing a new conjunction $M \vdash t'_1 \wedge \ldots \wedge M \vdash t'_l$ of deduction constraints for the immediate subterm $t'_j$ of $t_i$; we can label these $t_j$ with the respective subtrees of the derivation tree of $t_i$, so the resulting constraint still supports the interpretation $\mathcal{I}$.
– If the node is an application of the (*Decompose*) rule, then consider the ground term $t$ that is being decomposed in the derivation proof for $\mathcal{I}(t_i)$. We first consider different cases depending on how $t$ is derived:
  - If it is a composition step, then the intruder composed the term and then decomposed it subsequently. Since decomposition can only yield subterms of the composed term, one of the subtrees proves that the intruder can already derive $\mathcal{I}(t_i)$ and we can thus simplify the proof tree. We thus assume in the following that the proof tree contains no composition followed by a decomposition.
  - It cannot be an application of the (*Public*) rule, since that cannot have an analyzable subterm.
  - If $t$ is obtained by a decomposition step itself, then we regress to the respective term being decomposed, and we do so until we hit a term that is not obtained by decomposition. By the previous cases, this cannot be a composition step or public-constant step either, so all remains is following case:
  - The derivation of $t$ is a leaf, i.e., there is a $t' \in M$ such that $\mathcal{I}(t') = t$. We now show that in this case we can perform the decomposition step.

  Since decomposition is performed on $t$ in the derivation of $\mathcal{I}(t_i)$, we have that $\mathtt{Ana}(t) = (K, T)$ for some sets of ground terms $K$ and $T$, where $\mathcal{I}(M) \vdash k$ for every $k \in K$ and $\mathcal{I}(t_i) \in T$.

  We have two further cases, namely whether $t'$ (the term in $M$ whose instance is $t$) is a variable or not. If $t'$ is a variable, then again $t' = t_j$ for some $j < i$ and we can just replace the subtree for the derivation of $\mathcal{I}(t')$ with the derivation of $\mathcal{I}(t_j)$.

  Finally, if $t'$ is not a variable, then $\mathtt{Ana}(t') = (K', T')$ for some sets $K'$ and $T'$ with $\mathcal{I}(K') = K$ and $\mathcal{I}(P') = T$. Unless $T' \subseteq M$ (which is for instance the case if the decomposition step has already been applied previously, so we can simply replace the decomposition step with a leaf node), we can apply the decomposition rule of the constraint reduction and label the newly added conjuncts $M_i \vdash k'$ for every $k' \in K'$ with the respective derivation of $\mathcal{I}(k')$ of the previous proof tree. Thus, the resulting the constraint (also extending all $M_j$ that are supersets of $M_i$ with the terms from $P'$) supports the interpretation $\mathcal{I}$.

For termination, it is standard to define a *weight* $(n, m, l)$ for a constraint $\phi$, where

- $n$ is the number of free variables in $\phi$;
- $m$ is the number of unanalyzed subterms in the intruder knowledges of constraints, i.e., let $M_i^*$ be the set of all terms in $M_i$ and their subterms to which analysis has not yet been applied, and let $m = \Sigma_i |M_i^*|$, where $|M_i^*|$ is the number of unanalyzed terms in the set $M_i^*$ and $\Sigma_i |M_i^*|$ is the sum for all $M_i$ in a constraint;
- $l = \mathit{size}(\phi)$, where

$$\begin{array}{ll} \mathit{size}(\phi \wedge \phi') = \mathit{size}(\phi) + \mathit{size}(\phi') & \mathit{size}(M \vdash t) = \mathit{size}(t) \\ \mathit{size}(s \doteq t) = \mathit{size}(s) + \mathit{size}(t) & \mathit{size}(c) = \mathit{size}(x) = 1 \\ \mathit{size}(\neg\exists.\phi_\sigma) = 0 & \mathit{size}(f(t_1, \ldots, t_n)) = \mathit{size}(t_1) + \ldots + \mathit{size}(t_n) + 1 \end{array}$$

We order the components of this weight lexicographically, i.e., $(n, m, l) > (n', m', l')$ iff $n > n'$ or $(n = n'$ and $(m > m'$ or $(m = m'$ and $l > l')))$. Obviously, $>$ has no infinite descending chain. Now, for every derivation step $\phi \rightsquigarrow \phi'$

- either $\phi'$ has a smaller number of variables than $\phi$ ((*Unify*) or (*Equation*) with a substitution $\sigma \neq \mathit{identity}$), thus the $n$ component is smaller,
- or we apply (*Decompose*), marking the analyzed term (also in supersets of the respective knowledge) and thus decrease the $m$ component,
- or we apply any of the other rules without unification, so that the $l$ component decreases.

So, every $\rightsquigarrow$ step reduces the weight and termination then follows quite straightforwardly. □

**Lemma 3** *Let $\phi$ be a simple constraint where $\Gamma(s) = \Gamma(t)$ holds for all equations $s \doteq t$, and if the equation is under a negation (i.e., part of an inequality) then neither $s$ nor $t$ contain variables of composed types. Then, $\phi$ has a well-typed model, i.e., a well-typed interpretation $\mathcal{I}^T$ with $\mathcal{I}^T \models \phi$.*

*Proof.* For this proof, we first show that we can find a well-typed solution for all inequalities (with values in $\mathcal{C}_{pub}$ that the intruder can generate). Consider an inequalty $\phi = \neg\exists\bar{x}.\phi_\sigma$. Consider also a substitution $\theta$ of all free variables of $\phi$ (which are of atomic types) with constants of $\mathcal{C}_{pub}$ with corresponding types. Since $\phi$ is simple, the equations of $\theta(\phi_\sigma)$ cannot have a unifier. We have thus found a well-typed model for all the free variables of $\phi_\sigma$. It is straightforward to extend this to a well-typed model of the entire constraint; since all positive equations must be well-typed, and the intruder has an access to infinite reservoirs of fresh constants of all atomic types. □

**Theorem 4** *If a type-flaw-resistant protocol $P$ has an attack, then $P$ has a well-typed attack.*

*Proof.* The key idea is to consider a satisfiable constraint $\Phi = \phi \wedge tr_{M,E}(\Psi)$ that represents an attack against $P$, i.e., where $\phi$ is the constraint of a reachable state of $P$ and $tr_{M,E}(\Psi)$ is the translation of the violated goal $\Psi$ in that state. We have to show that the constraint has also a well-typed solution. By Theorem 3 and since $\Phi$ is satisfiable, we can use the symbolic intruder reduction rules to obtain a simple constraint $\Phi'$, i.e., $\Phi \rightsquigarrow^* \Phi'$. The point is now that for a type-flaw resistant protocol, all substitutions in this reduction must be well-typed. To prove that we have to show that if $P = (\mathcal{S}_0, \{\Psi_0, \cdots, \Psi_n\})$ is a type-flaw-resistant protocol and there exist a state $(\mathcal{S}; M; E; \phi)$ such that $(\mathcal{S}_0; \emptyset; \emptyset, \top) \Rightarrow^* (\mathcal{S}; M; E, \phi)$ and an interpretation $\mathcal{I}$ such that $\mathcal{I}, M, E \models_{\mathfrak{S}} \neg\Psi_i$ and $\mathcal{I} \models \phi$, then there exists a well-typed interpretation $\mathcal{I}^T$ such that $\mathcal{I}^T, M, E \models_{\mathfrak{S}} \neg\Psi_i$ and $\mathcal{I}^T \models \phi$. Recall that, by Theorem 1, $\mathcal{I}, M, E \models_{\mathfrak{S}} \neg\Psi_i$ iff $\mathcal{I} \models tr_{M,E}(\Psi_i)$, and $\mathcal{I}^T, M, E \models_{\mathfrak{S}} \neg\Psi_i$ iff $\mathcal{I}^T \models tr_{M,E}(\Psi_i)$.

Note that $\phi$ is initially $\top$, which is well-typed by default, and since $P$ is a type-flaw-resistant protocol, we have that: (1) all equations and events in the initial set of strands $\mathcal{S}_0$ are well-typed, (2) the type of each variable in the inequalities of the strands is atomic, and (3) $\Rightarrow$ does not introduce any negations (so an equality becomes an inequality or vice versa). Then, we have that all equations in $\phi$ are well-typed, and variables in inequalities have atomic types.

Recall that $\Psi_i = \forall\bar{x}.\psi \implies \psi_0$. We show now that $tr_{M,E}$ does not change these properties that originally hold in $\Psi_i$, i.e., $tr(\Psi_i)$ has the same properties that $\Psi_i$ has by the definition of type-flow-resistant protocol.

We show this by cases on $tr'_{M,E}$, since $tr(\cdot)$ passes $\Psi_i$ to $tr'_{M,E}(\cdot)$ with a negation on the $\psi_0$; that is fine since all equations (positive and negative) in $\Psi_i$ must be well-typed, and all variables that occur in such equations must have atomic types. Now we prove that $tr'_{M,E}$ preserves the above properties:

- $tr'_{M,E}(\mathsf{event}(t)) = \bigvee_{\mathsf{event}(s)\in E} s \doteq t$: all events in $E$ originate form the initial set of strands $\mathcal{S}_0$, and by $P$ being a type-flaw-resistant protocol, all events in $\mathcal{S}_0$ are well-typed, and thus $s \doteq t$ is well-typed.
- $tr'_{M,E}(\neg\mathsf{event}(t)) = \bigwedge_{\mathsf{event}(s)\in E} \neg s \doteq t$: we can conclude reasoning similarly to the previous cases where we already discussed the negation in $tr_{M,E}$.
- $tr'_{M,E}(s \doteq t) = s \doteq t$: immediate.
- $tr'_{M,E}(\exists\bar{x}.\psi) = \exists\bar{x}.tr'_{M,E}(\psi)$: immediate.
- $tr'_{M,E}(\neg s \doteq t) = \neg s \doteq t$: immediate.
- $tr'_{M,E}(\neg\exists\bar{x}.\,\mathsf{event}(t_1) \wedge \cdots \wedge \mathsf{event}(t_n) \wedge u_1 \doteq v_1 \wedge \cdots \wedge u_m \doteq v_m) = \bigwedge_{\mathsf{event}(s_1)\in E \ldots \mathsf{event}(s_n)\in E} \neg\exists\bar{x}.(s_1 \doteq t_1 \wedge \cdots \wedge t_n \doteq s_n \wedge u_1 \doteq v_1 \wedge \cdots \wedge u_m \doteq v_m)$: follows by reduction to previous cases.
- $tr'_{M,E}(\psi_1 \vee \psi_2) = tr'_{M,E}(\psi_1) \vee tr'_{M,E}(\psi_2)$ and the rest of the cases: follow by reduction to previous cases.

We need to prove that for a type-flaw-resistant protocol, if there is an interpretation $\mathcal{I} \models \Phi$, then there is a well-typed $\mathcal{I}^T \models \Phi$. By Theorem 3 and since $\Phi$ is satisfiable, we can use the symbolic intruder reduction rules to obtain a simple constraint $\Phi'$, i.e., $\Phi \rightsquigarrow^* \Phi'$.

Next we prove that all substitutions made in the reduction steps made until we reach a $\Phi'$ are well-typed substitutions. We prove this as follows.

Note that by definition of type-flaw-resistant protocols (Definition 10), all equations must be well-typed, and whenever a variable occurs in an equation in

$\Psi$ its type must be atomic; this also applies to inequalities in the strands of the type-flaw-resistant protocols. Moreover, all terms that can occur in a constraint are subterms of instances of terms in $SMP(P)$, i.e., when $t$ occurs in $\Phi$ then $t \sqsubseteq \vartheta(t')$ for some substitution $\vartheta$ and $t' \in SMP(P)$. We need to prove that all reductions made to $\Phi$ towards $\Phi'$ preserved the above properties. We prove that $tr'_{M,E}$ preserves the above properties as well. For this, we consider the two cases that involve substitutions in the constraint reduction rules, namely *(Unify)* and *(Equation)*. For the *(Unify)* rule, we proceed by cases of $s$ and $t$:

- If both $s$ and $t$ are atomic, then $s$ and $t$ cannot be variables, so the above property is preserved trivially, simply because they must be the same constant.
- If both are composed, then $\sigma(s) = \sigma(t)$ and there exist $u, v \in SMP$ and $\vartheta_1, \vartheta_2$ such that $\vartheta_1(u) = s$ and $\vartheta_2(v) = t$. Then, $\sigma(\vartheta_1(u)) = \sigma(\vartheta_2(v))$ and $\Gamma(u) = \Gamma(v) = \Gamma(s) = \Gamma(t)$ as the protocol is type-flaw-resistant, and so $\sigma$ is well-typed.
- If $t$ is variable, then it is simple and we proved earlier that if it has an ill-typed solution, then it also has a well-typed one.

For the *(Equation)* rule, we conclude immediately from the fact that $P$ is type-flaw-resistant; since all equations are well-typed, all unifications must be well-typed.

So far we have arrived at a $simple(\Phi')$ where $\Gamma(s) = \Gamma(t)$ holds for all equations $s \doteq t$, and if the equation is under a negation (i.e., part of an inequality), then neither $s$ nor $t$ contain variables of composed types. By Lemma 3, we conclude the existence of well-typed solution for such $\Phi'$. $\square$

**Theorem 5** *If two protocols $P_1$ and $P_2$ are parallel-composable and both $P_1$ and $P_2$ are secure in isolation in the typed model, then $P_1 \parallel P_2$ is secure (also in the untyped model).*

*Proof.* Consider an attack against $P_1 \parallel P_2$ violating a goal $\Psi$ of $P_1$. We show that the given attack works also on $P_1$ isolation, or one of the $P_i$ in isolation leaks one of the long-term secret constants. We use a similar argument as in Theorem 4: let $\phi$ be a constraint that represents the attack against $P_1 \parallel P_2$; we show how to extract a satisfiable constraint that represents either an attack against $P_1$ or $P_2$ in isolation, and that this attack works in the typed model.

First observe that composability of $P_1$ and $P_2$ implies that they are both type-flaw resistant. Thus, there is no unifier between two messages of $SMP(P_1)$ unless they have the same type, and the same holds for $SMP(P_2)$. Since also there are no unifiers between a message from $SMP(P_1)$ and a message from $SMP(P_2)$, we can derive that also $P_1 \parallel P_2$ is type-flaw resistant. By Theorem 4, if there is an attack against $P_1 \parallel P_2$, then there must be also a well-typed attack against $P_1 \parallel P_2$. We can thus without loss of generality assume that the given constraints that represent an attack against $P_1 \parallel P_2$ have a well-typed solution. This allows us first to get rid of all variables of composed types: we substitute each variable of a composed type $f(\tau_1, \ldots, \tau_n)$ by the expression $f(x_1, \ldots, x_n)$

where the $x_i$ are new variables of types $\tau_i$, and repeat this process until we have only variables of atomic types. Since the given constraint has a well-typed solution, so has the transformed one. Let $\mathcal{I}$ be such a well-typed solution.

As a second step, we substitute every variable $x$ of type private key with $\mathcal{I}(x)$.[7] Thus we have no variables of type private key anymore, and for every $\mathsf{pub}(t)$, $t$ is a public or secret long-term constant of type private key.

The next step in the proof is to introduce a label for each term and subterm that occurs in a $M \vdash t$ constraint in $\phi$, namely whether the corresponding term "belongs" to $P_1$ or to $P_2$. Recall that in $M \vdash t$, $t$ represents a message that the intruder sent to an honest agent (or was required to construct for violating a goal), and $M$ represents messages he has received from honest agents. By construction, all these terms are thus instances of either an $MP(P_1)$ or an $MP(P_2)$ term and can be labeled as such, with the exception of $\mathsf{pub}(t)$ terms in an intruder knowledge $M$ which would be available in both. Let us thus label all occurrences of $\mathsf{pub}(t)$ with label $\star$. For each term, we give the same label to all its subterms. Note that throughout the constraint, all occurrences of a variable thus receives the same labeling and we can thus speak of $P_1$-variables and $P_2$-variables. Similarly, all fresh constants $\mathcal{C}_{P_1}$ will be labeled $P_1$ and all fresh constants of $\mathcal{C}_{P_2}$ will be labeled $P_2$. However the public and private long-term constants may occur both with a $P_1$ and with a $P_2$ label; let us thus label them with $\star$ instead. In this way, all terms and their subterms have a consistent labeling in the sense that all occurrences of the term will bear the same label. Also by construction, in all equations $s \doteq t$, both $s$ and $t$ are labeled $P_1$ or both $P_2$.

As shown already in the proof of Theorem 4, during constraint reduction we obtain only terms that are instances of $SMP(P_1)$ or $SMP(P_2)$ or that are atomic. What we now show is that for one of the $P_i$ all those constraints $M \vdash t$ where $t$ is labeled $P_i$ can be solved using only messages in $M$ that are also labeled $P_i$ or $\star$ (and are not a long-term secret constant). This means that the constraint $M \vdash t$ can still be solved when removing from $M$ all messages labeled for the other protocol, so that we have an attack that works in the respective $P_i$ alone.

To that end, as in the proof of Theorem 4, we proceed along the well-formedness order of the constraint, solving the first non-simple one in a way that supports the given solution $\mathcal{I}$, and show that during this constraint reduction we never need to use a $P_2$ message to solve a $P_1$ constraint or vice versa. In particular, we will never perform a unification between a $P_1$-labeled and a $P_2$-labeled message.

Whenever applying the (*Equation*) rule on an equation $s \doteq t$, it is impossible that $s$ is labeled $P_1$ and $t$ is labeled $P_2$ (or vice versa) by construction, so also the resulting unifier produces equations with this property.

---

[7] The reader may wonder why we would not do this actually for all variables. In fact, we cannot, because in general the solution $\mathcal{I}$ may be exploiting that messages from $P_1$ can be used for $P_2$. Only for private keys, we know that they are either long-term secrets or long-term public values, either (supposedly) secret in both protocols or available in both.

More critical is the (*Unify*) rule, solving $M \vdash t$ by unifying $t$ with some term $s \in M$. Suppose $t$ is labeled $P_1$ or $\star$; $s$ may be labeled $P_1$, $P_2$ or $\star$. We show that there is a solution without using a $P_2$ message. Since (*Unify*) requires that $s, t \notin \mathcal{V}$, they either are both the same constant or they are both composed. We distinguish the following cases:

- If both are the same constant $c$, then we have any of the following sub-cases:
  - $c \in \mathcal{C}_{pub}$: then the constraint can rather be solved using the (*PubConsts*) rule instead (and thus there is not need to use any message from $P_2$).
  - $c \in \mathcal{C}_{priv}$: then $M$ contains a long-term secret constant. Let $P_i$ be the protocol from which $c$ was learned. Since all previous constraints (in the well-formedness order) are already simple, those that are labeled $P_i$ form a valid trace of $P_i$ alone that leaks $c$ and thus can be extended to an attack against $P_i$ (disregarding all further constraints).
  - $c \in \mathcal{C}_{P_1}$ or $c \in \mathcal{C}_{P_2}$, then by construction they are both labeled $P_1$ or both labeled $P_2$, so (*Unify*) does not destroy our invariant.
- If both are composed, then we further distinguish two cases:
  - $s = t = \mathsf{pub}(c)$: we have that $\mathsf{pub}(c)$ is available in both protocols by parallel-composability, i.e., in order to solve this in a $P_1$ constraint without using $P_2$, we can augment the attack trace with an initial step where the intruder learns $\mathsf{pub}(c)$ from the respective strand of $P_1$.
  - In all other cases, we can use again that all non-atomic messages are instances of terms in $SMP(P_1)$ or $SMP(P_2)$ and that the protocols are SMP-disjoint, so if $s$ and $t$ have a unifier, they must belong to the same $SMP(P_i)$.

Finally, consider analysis steps: in this case, when analyzing a message $m \in M$, we obtain constraints of the form $M \vdash k$ for some subterm $k$ of $m$. Moreover, we add then some subterm $p$ to $M$ in the analyzed constraint. Here, $k$ and $p$ must have the same label as $m$ or be labeled $\star$. It follows that to analyze a $P_1$-labeled message, we will never need to construct a $P_2$-labeled key (or vice versa).

In conclusion, we thus never need $P_1$ messages to solve a $P_2$ constraint or vice versa, with the exception of leaked long-term secret constants. In this case, however, we have already found an attack in an initial part of the constraint (in which all $P_1$ constraints can be solved without $P_2$ and vice versa). Thus, we obtain in all cases a sequence of constraints for the individual protocols, and one of them is an attack.