Typing and Compositionality for Security Protocols: A Generalization to the Geometric Fragment

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

[1] DTU Compute, Lyngby, Denmark
samo@dtu.dk
[2] School of Computing Science, Newcastle University, Newcastle upon Tyne, UK
[3] Department of Informatics, King's College London, 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* [4,6,13,18]. 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,

This work was partially supported by the EU FP7 Projects no. 318424, "FutureID: Shaping the Future of Electronic Identity" (futureid.eu), and by the PRIN 2010–2011 Project "Security Horizons". We thank Thomas Groß for many useful discussions.

G. Pernul et al. (Eds.): ESORICS 2015, Part II, LNCS 9327, pp. 209–229, 2015.
DOI: 10.1007/978-3-319-24177-7_11

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 [5,6].

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. Our contributions are 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 brief, 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, constraint reduction will never make an ill-typed substitution, where for compositionality "ill-typed" means to unify 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 [11]. 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.

Organization. In Sects. 2 and 3, we introduce a symbolic protocol model based on strands and properties in the geometric fragment. In Sect. 4, we reduce verification of the security properties to solving constraints. In Sects. 5 and 6, we give our typing and parallel compositionality results. In Sect. 7, we introduce a tool that checks if protocols are parallel-composable and report about first experimental results. In Sect. 8, we conclude and discuss related work. Proof sketches are in the appendix.

2 Messages, Formats and the Intruder Model

2.1 Messages

Let Σ be a finite set of *operators* (also referred to as *function symbols*); as a concrete example, Table 1 shows a Σ 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 Σ, $\mathcal{V}$ 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 σ.

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

Table 1. Example Operators Σ

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., 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., 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		$\mathtt{Ana}(t) = (\{\emptyset\},\{\emptyset\})$

TLS is $\mathbf{client\_hello(time, random, session\_id, cipher\_suites, comp\_methods)} = \mathsf{byte}(1) \cdot \mathsf{off}_3(\mathsf{byte}(3) \cdot \mathsf{byte}(3) \cdot \mathit{time} \cdot \mathit{random} \cdot \mathsf{off}_1(\mathit{session\_id}) \cdot \mathsf{off}_2(\mathit{cipher\_suites}) \cdot \mathsf{off}_1(\mathit{comp\_methods}))$, 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.

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 [19] that under certain conditions on formats this abstraction introduces neither false positives nor false negatives. The conditions are essentially that formats must be parsed in an unambiguous way and must be pairwise disjoint; then every attack on the concrete bytestring model can be simulated in the model based on abstract format symbols (in the free algebra). 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*:*

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

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

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 Σ (where n denotes the arity of f). Table 1 provides an example Σ 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 [12]) 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 ϕ 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 ϕ_σ defines *equations* on messages, and we can existentially quantify variables in them, e.g., consider a ϕ 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 ϕ).

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 ϕ, the model relation $\mathcal{I} \models \phi$ is:*

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

We say that $\mathcal{I}$ is a model *of ϕ iff $\mathcal{I} \models \phi$, and that ϕ 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 ϕ is* closed*, in symbols closed(ϕ), iff $fv(\phi) = \emptyset$.*

Every constraint ϕ 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 ϕ_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 ϕ_i satisfies the following condition: we can order the conjuncts of ϕ_i such that $\phi_i = M_1 \vdash t_1 \wedge \ldots \wedge M_n \vdash t_n \wedge \phi_i'$, where ϕ_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 [21], 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 ϕ_σ 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[1] $\mathcal{S}$ of closed operational strands, a finite set M of terms representing the intruder knowledge, a finite set E of events, and a formula ϕ. $fv(\cdot)$ 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 ϕ 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 ϕ 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 α-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 ϕ' to the constraint ϕ. 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.

[1] 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.

3.3 Goal Predicates in the Geometric Fragment

We express goals by *state formulas in the geometric fragment* [11]. Here, we also allow to directly talk about the intruder knowledge, but in a restricted manner so that we obtain constraints of the form ϕ. Security then means: every reachable 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 ϕ 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 Ψ 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* Ψ*, 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:*

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

Definition 5. *A protocol* $P = (\mathcal{S}_0, \{\Psi_0, \cdots, \Psi_n\})$*, where the* Ψ_i *are closed state formulas, has an attack against goal* Ψ_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* $\mathcal{I} \models_{\mathfrak{S}} \phi$*. 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 Ψ_i is closed.

Example 1. If a protocol generates the event[2] $\mathsf{secret}(x_A, x_B, x_m)$ 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 i denotes the intruder. The release event can be used to model declassification of secrets as needed

[2] 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.

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* [15] 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: $\forall x_A x_B x_m\, x_C x'_C.\, \mathsf{running}(x_A, x_B, x_m, x_C) \wedge \mathsf{running}(x_A, x_B, x_m, x'_C) \implies x_A = \mathsf{i} \vee x_B = \mathsf{i} \vee x_C = x'_C$. □

4 Constraint Solving

We first show how to translate every state formula Ψ in the geometric fragment for a given symbolic state $\mathfrak{S} = (\mathcal{S}; M; E; \phi)$ into a constraint ϕ' (in the fragment defined in Sect. 3.1) so that the models of $\phi \wedge \phi'$ represent exactly all concrete instances of $\mathfrak{S}$ that violate Ψ. Then, we extend a rule-based procedure to solve ϕ-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 Ψ. As mentioned earlier, we require that Ψ is closed. Let us further assume that the bound variables of Ψ are disjoint from the variables (bound or free) of $\mathcal{S}$, M, E, and ϕ. We now define a *translation function* $tr_{M,E}(\Psi) = \phi'$ where ϕ' represents the negation of Ψ with respect to intruder knowledge M and events E. The negation is actually manifested in the first line of the definition:

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

Theorem 1. *Let $\mathfrak{S} = (\mathcal{S}; M; E; \phi)$ be a symbolic state and Ψ 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 ϕ 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 ϕ_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 [7,16,18,20], 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 $\dfrac{\phi'}{\phi}$ expressing that ϕ' entails ϕ (if the side condition *cond* holds). However, we will usually read the rule backwards, i.e., as: one way to solve ϕ is ϕ'.

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

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

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

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

where $(M' \vdash t)^{T \gg M}$ *is* $M' \cup T \vdash t$ *if* $M \subseteq M'$ *and* $M' \vdash t$ *otherwise,* $(\cdot)^{T \gg M}$ *extends as expected,* $eq(\sigma) = x_1 \doteq t_1 \wedge \ldots \wedge x_n \doteq t_n$ *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 (*mgu* extends as expected). 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 σ; this is done by applying σ 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 that we can easily check them for satisfiability.

Definition 7. *A constraint ϕ is* simple*, written simple(ϕ), iff $\phi = \phi_1 \wedge \ldots \wedge \phi_n$ such that for each ϕ_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 ϕ;*
- *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 ϕ_i and $\bar{c}$ fresh constants that do not appear in ϕ.*

Theorem 2. *If simple(ϕ), then ϕ is satisfiable.*

Theorem 3 (Adaption of *[18, 20]*). *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 τ. Based on the type information of atomic terms, we define the *typing function* Γ 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 Γ 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 σ 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 β_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 σ 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 Sect. 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 α-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 Ψ *are defined as follows:*

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

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 σ, 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 any variable x that occurs in equations or events of G, $\Gamma(x) \in \mathfrak{T}_a$.*
- *For any 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 as $\mathsf{scrypt}(k, \mathsf{f}_1(x_1, y_1))$ 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. 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 Sect. 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* β_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 they 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 $\mathcal{C}_{priv}^{\beta_i}$, 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 (available at http://www2.compute.dtu.dk/~samo/APCC.zip), 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.

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 (e.g., [4,6,13, 18]) and on parallel protocol composition (e.g., [2,8–10,12]) 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 make several generalizations over previous results. First, we are not limited to a fixed set of properties like secrecy [3,6]. Instead, we consider the entire geometric fragment proposed by Guttman [11] that we believe is the most expressive language that can work with the given constraint-solving argument that is at the core of handling typing and compositionality results uniformly. Other expressive property languages have been considered, e.g., PS-LTL for typing results [4]; an in-depth comparison of the various existing property languages and their relative expressiveness is yet outstanding. Another common limitation is to rely on a fixed public key infrastructure, e.g., [3,4,8]. Our approach in contrast allows for the exchange of public keys (including freshly generated ones). Moreover, early works on typing and parallel composition used a fixed tagging scheme, whereas we use the more general notion of non-unifiable subterms for messages that have different meaning. Using the notion of formats, our results are applicable to existing real-world protocols like TLS with their actual formats.

Our work considered so far protocols only in the initial term algebra without any algebraic properties. There are some promising results for such properties (e.g., [9,14,17]) 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 [9], where one protocol establishes a key that is used by another protocol as input.

A Appendix: Proofs of the Technical Results

For space reasons, we only outline the proofs to convey the main ideas. The detailed proofs are in [1].

Proof Sketch of Theorem 1. We prove by induction a corresponding property for tr' that is invoked by the tr function, i.e., that $\mathcal{I}, M, E \models_{\mathfrak{S}} \psi$ iff $\mathcal{I} \models tr'_{M,E}(\psi)$ for all suitable $\mathcal{I}, M, E$ and ψ. Here we use the fact that in all reachable states, E is finite, i.e., $\mathsf{event}(t) \in E$ is equivalent to a finite enumeration $t = e_1 \vee \ldots \vee t = e_n$. All other cases and the extension to tr are straightforward. The well-formedness follows from the fact that in each state, the knowledge M is a superset of all M' that occur in a deduction constraint $M' \vdash t$ in ϕ. □

Proof Sketch of Theorem 2. Since ϕ is simple, it 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 ϕ, as well as 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 ϕ 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 equations are obviously satisfiable. It remains to show that the inequalities are satisfiable under θ. 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, and since ϕ is simple we have $mgu(\theta(\phi_\sigma)) = \emptyset$. Thus, $\theta(\phi_0)$ 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 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 ϕ. Then ϕ is satisfiable iff $\theta(\phi)$ is satisfiable. Moreover, ϕ is satisfiable iff $mgu(\theta(\phi_\sigma)) = \emptyset$.*

Proof Sketch. If $\theta(\phi)$ is unsatisfiable, then also ϕ is unsatisfiable. For the other direction, we show that the following two formulas are a contradiction: (1) $\exists\bar{y}.\forall\bar{x}.\bigvee_{i=1}^{n} s_i \neq t_i$ and (2) $\exists\bar{x}.\bigwedge_{i=1}^{n} \theta(s_i) = \theta(t_i)$. 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 θ and ξ are substitutions with disjoint domain and grounding, we have $\theta(\xi(\cdot)) = \xi(\theta(\cdot))$, and thus we obtain (2') $\bigwedge_{i=1}^{n} \theta(\xi(s_i)) = \theta(\xi(t_i))$. By (1), choosing a particular value for the $\bar{x}$, we obtain (1') $\exists\bar{y}.\bigvee_{i=1}^{n} \xi(s_i) \neq \xi(t_i)$. 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 (1") $\exists \bar{y}.\, s \neq t$ and (2") $\theta(s) = \theta(t)$. By case distinction on whether s and t are atomic or not, follows immediately that (1") and (2") are a contradiction. Now we can decide the satisfiability of ϕ with the *mgu*-algorithm since $\theta(\phi)$ is a closed formula and the remaining variables of $\theta(\phi_\sigma)$ are the $\bar{x}$. □

Proof Sketch of Theorem 3. As this is building on standard results [18,20], we only describe the basic idea and highlight the particularities of our adaption. Let us write $\phi \rightsquigarrow \phi'$ if $\frac{\phi'}{\phi}$ is an instance of a reduction rule, i.e., representing one solution step. It is straightforward that the rules are sound, i.e., that $\mathcal{I} \models \phi'$ and $\phi \rightsquigarrow \phi'$ imply $\mathcal{I} \models \phi$. The hard part is the completeness, i.e., when $\mathcal{I} \models \phi$, then either ϕ is already simple or we can apply some rule, obtaining $\phi \rightsquigarrow \phi'$ for some ϕ' 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). The core idea is as follows. Since $\mathcal{I}$ satisfies ϕ, for every intruder deduction $M \vdash t$ in ϕ, there exists a proof $\mathcal{I}(M) \vdash \mathcal{I}(t)$ according to 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 ϕ) 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 ϕ' 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.

For the equalities $s \doteq t$ that are not yet simple (i.e., where neither s nor t is a variable that occurs only once in the constraint), we can at any time in the reduction apply (*Equation*), if the equation actually *has* a unifier. If not, obviously the entire constraint is satisfiable. For the inequalities, suppose we have a non-simple inequality $\phi_0 = \neg\exists \bar{x}.\phi_\sigma$, i.e., $mgu(\theta(\phi_\sigma)) \neq \emptyset$ for a substitution θ from the free variables of ϕ_0 to fresh constants. Then, by Lemma 2, ϕ_0 is not satisfiable, contradicting that $\mathcal{I} \models \phi$. This concludes the completeness proof.

For termination, it is standard to define a *weight* (n, m, l) for a constraint ϕ, where n is the number of free variables in ϕ, m is the number of unanalyzed subterms in the intruder knowledges of constraints; and l is the size of the constraint (in symbols). Ordering these three components lexicographically, every $\rightsquigarrow$ step reduces the weight. □

Proof Sketch of Theorem 4. 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 ϕ is the constraint of a reachable state of P and $tr_{M,E}(\Psi)$ is the translation of the violated goal in that state. We have to show that the constraint has also a well-typed solution. By Theorem 3 and since Φ is satisfiable, we can use the symbolic intruder reduction rules to obtain a simple constraint Φ', 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 see that, first observe that by construction, all equations $s \doteq t$ (including inequalities) of Φ have $\Gamma(s) = \Gamma(t)$ and for inequalities, we have that all variables of s and t are of atomic type. Further, all terms that occur are either

instances of terms in $SMP(P)$ or atomic. We have to show also that during all reduction steps, this property is preserved. Now note that we cannot apply the (*Unify*) rule on a pair of terms s and t that are variables. So, they are either both the same constant (which trivially preserves our invariants) or they are composed and thus instances of terms in $SMP(P)$. That in turn means that s and t can only have a unifier if $\Gamma(s) = \Gamma(t)$ by the type-flaw resistance property. Thus, the resulting unifier is well-typed. For the (*Equation*) rule, we can obtain only well-typed unifiers since all equations have to be well-typed. Finally, with a similar construction as in Theorem 2 and using the fact that inequalities cannot have composed variables, we can show that, when reaching a simple constraint (in which all equations are well-typed), it has a well-typed solution. □

Proof Sketch of Theorem 5. Consider an attack against $P_1 \parallel P_2$ violating a goal Ψ of P_1. By Theorem 4, there must be a well-typed attack against $P_1 \parallel P_2$, so we consider only a well-typed attack. We show that this attack works also in P_1 isolation, or one of the P_i in isolation leaks one of the long-term secret constants. We use again a similar argument as in Theorem 4: let ϕ be a constraint that represents the attack against $P_1 \parallel P_2$ and thus has a well-typed model $\mathcal{I}$; we can then extract a constraint that represents an attack against P_1 or P_2 in isolation. First, it does thus not change satisfiability of ϕ if we substitute each variable of a composed type $f(\tau_1, \ldots, \tau_n)$ by an expression $f(x_1, \ldots, x_n)$ where x_i are new variables of types τ_i, and repeat this process until we have only variables of atomic types. We substitute every variable x of type private key with $\mathcal{I}(x)$. 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.

For every constraint $M \vdash t$ in ϕ, t is a message that was received by a strand of either P_1 or P_2 and each $s \in M$ is a message sent by a strand of either P_1 or P_2. It is thus just a matter of book keeping to label t and each element of M with either P_1 or P_2 accordingly. The property of parallel-composable requires that all public constants of $\mathcal{C}_{pub}$ and all public keys $\mathsf{pub}(c)$ for c of type private key are available to the intruder in each of the protocols; so when they occur in the knowledge M of a constraint, we shall label them with $\star$ instead, as they are not protocol-specific. By construction, no variable can occur both in a P_1-labeled term and in a P_2-labeled term (and this property is preserved over all reductions); we can thus also label variables uniquely as being either P_1-variables or P_2-variables. By construction, in all $s \doteq t$, both s and t are labeled the same.

Next, we can attack one of the two protocols without a unification between a P_1 and a P_2 message. In the constraint system, this means that a constraint $M \vdash t$ where t is labeled P_1 can always be solved when removing all P_2-labeled messages from M. Following the reductions for a given well-typed solution $\mathcal{I}$, the only critical rule is (*Unify*), solving $M \vdash t$ by unifying t with some term $s \in M$. Suppose t is labeled P_1; 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. In case of a constant, it cannot belong to the protocol-specific constants $\mathcal{C}_{P_1}$ or $\mathcal{C}_{P_2}$ (since they are disjoint by construction labeled for the respective protocol), so it

must be a long-term constant, belonging either to $\mathcal{C}_{pub}$ (then we can solve this constraint instead with the (*PubConsts*) rule), or to $\mathcal{C}_{priv}$. In the latter case, we have that one of the two protocols has leaked a long-term secret, and we can extract the constraints up to this point as a witness that we have already an attack for one protocol in isolation. It remains the case of composed messages s and t being unified. One special case is that $s = t = \mathsf{pub}(c)$, in which case we have that $\mathsf{pub}(c)$ is available in both protocols by parallel-composable. 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 $SMP(P_i)$. Thus the attack never relies on the unification between a P_1 and a P_2 message, and we can extract a pure P_1 or a pure P_2 constraint that violates a goal of the respective protocol. Note that it is either a violation of a long-term secrecy goal or violating the goal Ψ of P_1 that the initial attack against $P_1 \parallel P_2$ violates. □

References

1. Almousa, O., Mödersheim, S., Modesti, P., Viganò, L.: Typing and compositionality for security protocols: a generalization to the geometric fragment (extended version). DTU Compute Technical report-2015-03 (2015). http://www.imm.dtu.dk/~samo/
2. Andova, S., Cremers, C.J.F., Gjøsteen, K., Mauw, S., Mjølsnes, S.F., Radomirovic, S.: A framework for compositional verification of security protocols. Inf. Comput. **206**(2–4), 425–459 (2008)
3. Arapinis, M., Duflot, M.: Bounding messages for free in security protocols. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007. LNCS, vol. 4855, pp. 376–387. Springer, Heidelberg (2007)
4. Arapinis, M., Duflot, M.: Bounding messages for free in security protocols - extension to various security properties. Inf. Comput. **239**, 182–215 (2014)
5. Armando, A., Compagna, L.: SATMC: a sat-based model checker for security protocols. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 730–733. Springer, Heidelberg (2004)
6. Blanchet, B., Podelski, A.: Verification of cryptographic protocols: tagging enforces termination. Theor. Comput. Sci. **333**(1–2), 67–90 (2005)
7. Comon-Lundh, H., Delaune, S., Millen, J.K.: 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)
8. Cortier, V., Delaune, S.: Safely composing security protocols. Form. Methods Syst. Des. **34**, 1–36 (2009)
9. Ciobâcă, Ş., Cortier, V.: Protocol composition for arbitrary primitives. In: CSF, pp. 322–336. IEEE (2010)
10. Guttman, J.D.: Cryptographic protocol composition via the authentication tests. In: de Alfaro, L. (ed.) FOSSACS 2009. LNCS, vol. 5504, pp. 303–317. Springer, Heidelberg (2009)
11. Guttman, J.D.: Establishing and preserving protocol security goals. J. Comput. Secur. **22**(2), 203–267 (2014)
12. Guttman, J.D., Thayer, F.J.: Protocol independence through disjoint encryption. In: CSFW, pp. 24–34. IEEE (2000)

13. Heather, J., Lowe, G., Schneider, S.: How to prevent type flaw attacks on security protocols. J. Comput. Secur. **11**(2), 217–244 (2003)
14. Küsters, R., Truderung, T.: Using ProVerif to analyze protocols with Diffie-Hellman exponentiation. In: CSF, pp. 157–171. IEEE (2009)
15. Lowe, G.: A hierarchy of authentication specifications. In: CSFW, pp. 31–44 (1997)
16. Millen, J.K., Shmatikov, V.: Constraint solving for bounded-process cryptographic protocol analysis. In: CCS, pp. 166–175. ACM (2001)
17. Mödersheim, S.: Diffie-Hellman without difficulty. In: Barthe, G., Datta, A., Etalle, S. (eds.) FAST 2011. LNCS, vol. 7140, pp. 214–229. Springer, Heidelberg (2012)
18. Mödersheim, S.: Deciding security for a fragment of ASLan. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 127–144. Springer, Heidelberg (2012)
19. Mödersheim, S., Katsoris, G.: A sound abstraction of the parsing problem. In: CSF, pp. 259–273. IEEE (2014)
20. Rusinowitch, M., Turuani, M.: Protocol insecurity with a finite number of sessions and composed keys is NP-complete. Theor. Comput. Sci. **299**(1–3), 451–475 (2003)
21. Thayer, F.J., Herzog, J.C., Guttman, J.D.: Strand spaces: proving security protocols correct. J. Comput. Secur. **7**(1), 191–230 (1999)