Abstractions for Security Protocol Verification

Binh Thanh Nguyen and Christoph Sprenger

Institute of Information Security, ETH Zurich, Switzerland
{thannguy,sprenger}@inf.ethz.ch

Abstract. We present a large class of security protocol abstractions with the aim of improving the scope and efficiency of verification tools. We propose typed abstractions, which transform a term's structure based on its type, and untyped abstractions, which remove atomic messages, variables, and redundant terms. Our theory improves on previous work by supporting a useful subclass of *shallow* subterm-convergent rewrite theories, user-defined types, and untyped variables to cover type flaw attacks. We prove soundness results for an expressive property language that includes secrecy and authentication. Applying our abstractions to realistic IETF protocol models, we achieve dramatic speedups and extend the scope of several modern security protocol analyzers.

1 Introduction

Security protocols play a central role in today's networked applications. Past experience has amply shown that informal arguments justifying the security of such protocols are insufficient. This makes security protocols prime candidates for formal verification. In the last two decades, research in formal security protocol verification has made enormous progress, which is reflected in many state-of-the-art tools including AVANTSSAR [1], ProVerif [6], Maude-NPA [14], Scyther [10], and Tamarin [22]. These tools can verify small to medium-sized protocols in a few seconds or less, sometimes for an unbounded number of sessions. Despite this success, they can still be challenged when verifying real-world protocols such as those defined in standards and deployed on the internet (e.g., TLS, IKE, and ISO/IEC 9798). Such protocols typically have messages with numerous fields, support many alternatives (e.g., cryptographic setups), and may be composed from more basic protocols (e.g., IKEv2-EAP).

Abstraction [7] is a standard technique to over-approximate complex systems by simpler ones for verification. Sound abstractions preserve counterexamples (or attacks in security terms) from concrete to abstracted systems. In the context of security protocols, abstractions are extensively used. Here, we only mention a few examples. First, the Dolev-Yao model is a standard (not necessarily sound) abstraction of cryptography. Second, many tools use abstractions to map the verification problem into the formalism of an efficient solver or reasoner. We call these *back-end* abstractions. For example, ProVerif [6] translates models in the applied pi calculus to a set of Horn clauses, SATMC [4] reduces protocol verification to SAT solving, and Paulson [25] models protocols as inductively

defined trace sets. Finally, some abstractions aim at speeding up automated analysis by simplifying protocols within a given protocol model before feeding them to verifiers [18,23]. Our work belongs to this class of *front-end* abstractions.

Extending Hui and Lowe's work [18], we proposed in [23] a rich class of protocol abstractions and proved its soundness for a wide range of security properties. We used a type system to uniformly transform all terms of a given type (e.g., a pattern in a protocol role and its instances during execution) whereas [18] only covers ground terms. Our work [23] exhibits several limitations: (1) the theory is limited to the free algebra over a fixed signature; (2) all variables have strict (possibly structured) types, hence we cannot precisely model ticket forwarding or Diffie-Hellman exchanges. While the type system enables fine-grained control over abstractions (e.g., by discerning different nonces), it may eliminate realistic attacks such as type flaw attacks; (3) some soundness conditions involving quantifiers are hard to check in practice; and (4) it presents few experimental results for a single tool (SATMC) using abstractions that are crafted manually.

In this work, we address all the limitations above. First, we work with a useful subclass of *shallow* subterm-convergent rewrite theories modulo a set of axioms to model cryptographic operations. Second, we support untyped variables, user-defined types, and subtyping. User-defined types enable the grouping of similar atomic types (e.g., session keys) and adjusting the granularity of matching in message abstraction. Third, we have separated the removal of variables, atomic messages, and redundancies (new untyped abstractions) from the transformation of the message structure (typed abstractions). This simplifies the specifications and soundness proof of typed abstractions. Fourth, we provide effectively checkable syntactic criteria for the conditions of the soundness theorem. Finally, we extended Scyther [10] with fully automated support for our abstraction methodology. We validated our approach on an extensive set of realistic case studies drawn from the IKEv1, IKEv2, ISO/IEC 9798, and PANA-AKA standard proposals. Our abstractions result in very substantial performance gains. We have also obtained positive results for several other state-of-the-art verifiers (ProVerif, CL-Atse, OFMC, and SATMC) with manually produced abstractions.

Example: The IKEv2-mac protocol The Internet Key Exchange (IKE) family of protocols is part of the IPsec protocol suite for securing Internet Protocol (IP) communication. IKE establishes a shared key, which is later used for securing IP packets, realizes mutual authentication, and offers identity protection as an option. Its first version (IKEv1) dates back to 1998 [17]. The second version (IKEv2) [20] significantly simplifies the first one. However, the protocols in this family are still complex and contain a large number of fields.

Concrete protocol. As our running example, we present a member of the IKEv2 family, called IKEv2-mac (or $\mathsf{IKE_m}$ for short), which sets up a session key using a Diffie-Hellman (DH) key exchange, provides mutual authentication based on MACs, and also offers identity protection. We use Cremers' models of IKE [11] as a basis for our presentation and experiments (see Section 4.2). Our starting point

is the following concrete $\mathsf{IKE_m}$ protocol between an initiator A and a responder B.

$$\begin{aligned}
&\mathsf{IKE_m}(1).\ \ A \to B : SPIa, o, sA1, g^x, Na\\
&\mathsf{IKE_m}(2).\ \ B \to A : SPIa, SPIb, sA1, g^y, Nb\\
&\mathsf{IKE_m}(3).\ \ A \to B : SPIa, SPIb, \{\!|A, B, AUTHa, sA2, tSa, tSb|\!\}_{SK}\\
&\mathsf{IKE_m}(4).\ \ B \to A : SPIa, SPIb, \{\!|B, AUTHb, sA2, tSa, tSb|\!\}_{SK}
\end{aligned}$$

Here, *SPIa* and *SPIb* denote the *Security Parameter Indices* that determine cryptographic algorithms, o is a constant number, *sA1* and *sA2* are *Security Associations*, g is the DH group generator, x and y are secret DH exponents, *Na* and *Nb* are nonces, and *tSa* and *tSb* denote *Traffic Selectors* specifying certain IP parameters. $AUTHa$ and $AUTHb$ denote the authenticators of A and B and SK the session key derived from the DH key g^{xy}. These are defined as follows.

$$\begin{aligned}
SK &= \mathsf{kdf}(Na, Nb, g^{xy}, SPIa, SPIb)\\
AUTHa &= \mathsf{mac}(\mathsf{sh}(A, B), SPIa, o, sA1, g^x, Na, Nb, \mathsf{prf}(SK, A))\\
AUTHb &= \mathsf{mac}(\mathsf{sh}(B, A), SPIa, SPIb, sA1, g^y, Nb, Na, \mathsf{prf}(SK, B))
\end{aligned}$$

We model the functions mac, kdf, and prf as hash functions and use $\mathsf{sh}(A, B)$ and $\mathsf{sh}(B, A)$ to refer to the (single) long-term symmetric key shared by A and B.

We consider the following security properties: (P1) the secrecy of the DH key g^{xy}, which implies the secrecy of SK, and (P2) mutual non-injective agreement on the nonces *Na* and *Nb* and the DH half-keys g^x and g^y.

Abstraction. Our theory supports the construction of abstract models by removing inessential fields and operations. For example, in $\mathsf{IKE_m}$ we can remove: (i) the symmetric encryptions with the session key SK; then (ii) all atomic top-level fields except *Na* and *Nb*; (iii) all fields of SK except the DH key g^{xy}; and (iv) from the authenticators: the fields *SPIa*, *SPIb*, and *sA1* and the application of prf including the agent names underneath. The resulting protocol is $\mathsf{IKE_m^2}$:

$$\begin{aligned}
&\mathsf{IKE_m^2}(1).\ \ A \to B : g^x, Na && \mathsf{IKE_m^2}(3).\ \ A \to B : AUTHa\\
&\mathsf{IKE_m^2}(2).\ \ B \to A : g^y, Nb && \mathsf{IKE_m^2}(4).\ \ B \to A : AUTHb
\end{aligned}$$

where $SK = \mathsf{kdf}(g^{xy})$ and $AUTHa = \mathsf{mac}(\mathsf{sh}(A, B), o, g^x, Na, Nb, SK)$ for role A and $AUTHb = \mathsf{mac}(\mathsf{sh}(B, A), g^y, Nb, Na, SK)$ for role B.

Scyther verifies the properties (P1) and (P2) in 8.7s on the concrete and in 1.7s on an automatically generated abstract protocol (which is less intuitive than the one presented here). Our soundness results imply that the original protocol $\mathsf{IKE_m}$ also enjoys these properties. We chose the protocol $\mathsf{IKE_m}$ as running example for its relative simplicity compared to the other protocols in our case studies. In many of our experiments (Section 4.2), our abstractions (i) result in much more substantial speedups, or (ii) enable the successful unbounded verification of a protocol where it times out or exhausts memory on the original protocol.

2 Security protocol model

We define a term algebra $\mathcal{T}_\Sigma(V)$ over a signature Σ and a set of variables V in the standard way. Let Σ^n denote the symbols of arity n. We call the elements

of Σ^0 *atoms* and write $\Sigma^{\geq 1}$ for the set of proper function symbols. For a fixed $\Sigma^{\geq 1}$, we will vary Σ^0 to generate different sets of terms, denoted by $\mathcal{T}(V, \Sigma^0)$, including terms in protocol roles, network messages, and types. We write $\mathit{subs}(t)$ for the set of subterms of t and define the size of t by $|t| = |\mathit{subs}(t)|$. We also define $\mathit{vars}(t) = \mathit{subs}(t) \cap V$. If $\mathit{vars}(t) = \emptyset$ then t is called *ground*. We denote the top-level symbol of a (non-variable) term t by $\mathit{top}(t)$ and the set of its symbols in $\Sigma^{\geq 1}$ by $\mathit{ct}(t)$. A position is a sequence of natural numbers. We denote the subterm of t at position p with $t|_p$ and write $t[u]_p$ for the term obtained by replacing $t|_p$ at position p by u. We also partition Σ into sets of public and private symbols, denoted by Σ_{pub} and Σ_{pri}. We assume Σ_{pub} includes pairing $\langle \cdot, \cdot \rangle$ which associates to the right, e.g., $\langle t, u, v \rangle = \langle t, \langle u, v \rangle \rangle$. We usually omit the tuple brackets under other symbols, e.g., we write $\{\!|t, u, v|\!\}_k$ rather than $\{\!|\langle t, u, v \rangle|\!\}_k$. We define the *splitting* function by $\mathit{split}(\langle t, u \rangle) = \mathit{split}(t) \cup \mathit{split}(u)$ on pairs and $\mathit{split}(t) = \{t\}$ on other terms t. We call the elements of $\mathit{split}(t)$ the *fields* of t. For $n \in \mathbb{N}$, we write $\widetilde{n}$ to denote $\{1, \ldots, n\}$.

The set of *message terms* is $\mathcal{M} = \mathcal{T}(\mathcal{V}, \mathcal{A} \cup \mathcal{F} \cup \mathcal{C})$, where $\mathcal{V}$, $\mathcal{A}$, $\mathcal{F}$, and $\mathcal{C}$ are infinite and pairwise disjoint sets of variables, agents, fresh values, and constants. We partition $\mathcal{A}$ into sets of honest and compromised agents: $\mathcal{A} = \mathcal{A}_H \cup \mathcal{A}_C$.

2.1 Type system

We introduce a type system akin to [2] and extend it with subtyping. We define the set of atomic types by $\mathcal{Y}_{at} = \mathcal{Y}_0 \cup \{\alpha, \mathit{msg}\} \cup \{\beta_n \mid n \in \mathcal{F}\} \cup \{\gamma_c \mid c \in \mathcal{C}\}$, where α, β_n, and γ_c are the types of agents, the fresh value n, and the constant c, respectively. Moreover, *msg* is the type of all messages and $\mathcal{Y}_0$ is a disjoint set of user-defined types. The set of all types is then defined by $\mathcal{Y} = \mathcal{T}(\emptyset, \mathcal{Y}_{at})$.

We assume that all variables have an atomic type, i.e., $\mathcal{V} = \{\mathcal{V}_\tau\}_{\tau \in \mathcal{Y}_{at}}$ is a family of disjoint infinite sets of variables. Let $\Gamma : \mathcal{V} \to \mathcal{Y}_{at}$ be such that $\Gamma(X) = \tau$ iff $X \in \mathcal{V}_\tau$. We extend Γ to atoms by defining $\Gamma(a) = \alpha$, $\Gamma(n) = \beta_n$, and $\Gamma(c) = \gamma_c$ for $a \in \mathcal{A}$, $n \in \mathcal{F}$, and $c \in \mathcal{C}$, and then homomorphically to all terms $t \in \mathcal{M}$. We call $\tau = \Gamma(t)$ the *type of* t and sometimes also write $t : \tau$.

The subtyping relation $\preccurlyeq$ on types is defined by the following inference rules and by two additional rules (not shown) defining its reflexivity and transitivity.

$$\frac{\tau \in \mathcal{Y}}{\tau \preccurlyeq \mathit{msg}}\ \mathrm{S}(\mathit{msg}) \qquad \frac{\tau_1 \preccurlyeq_0 \tau_2}{\tau_1 \preccurlyeq \tau_2}\ \mathrm{S}(\preccurlyeq_0) \qquad \frac{\tau_1 \preccurlyeq \tau_1' \quad \cdots \quad \tau_n \preccurlyeq \tau_n'}{c(\tau_1, \ldots, \tau_n) \preccurlyeq c(\tau_1', \ldots, \tau_n')}\ \mathrm{S}(c \in \Sigma^n)$$

Every type is a subtype of *msg* by the first rule. The second rule embeds a user-defined *atomic subtyping relation* $\preccurlyeq_0 \subseteq (\mathcal{Y}_{at} \setminus \{\mathit{msg}\}) \times \mathcal{Y}_0$, which relates atomic types (except *msg*) to user-defined atomic types in $\mathcal{Y}_0$. For simplicity, we require that $\preccurlyeq_0$ is a partial function. The third rule ensures that subtyping is preserved by all symbols. The set of subtypes of τ is $\tau{\downarrow} = \{\tau' \in \mathcal{Y} \mid \tau' \preccurlyeq \tau\}$.

2.2 Equational theories

An *equation* over a signature Σ is an unordered pair $\{s, t\}$, written $s \simeq t$, where $s, t \in \mathcal{T}_\Sigma(\mathcal{V}_{\mathit{msg}})$. An *equation presentation* $\mathcal{E} = (\Sigma, E)$ consists of a signature

Σ and a set E of equations over Σ. The *equational theory* induced by $\mathcal{E}$ is the smallest Σ-congruence, written $=_E$, containing all instances of equations in E. We often identify $\mathcal{E}$ with the induced equational theory.

A *rewrite rule* is an oriented pair $l \to r$, where $vars(r) \subseteq vars(l) \subseteq \mathcal{V}_{msg}$. A *rewrite theory* is a triple $\mathcal{R} = (\Sigma, Ax, R)$ where Σ is a signature, Ax a set of Σ-equations, and R a set of rewrite rules. The rewriting relation $\to_{R,Ax}$ on $\mathcal{T}_\Sigma(V)$ is defined by $t \to_{R,Ax} t'$ iff there exists a non-variable position p in t, a rule $l \to r \in R$, and a substitution σ such that $t|_p =_{Ax} l\sigma$ and $t' = t[r\sigma]_p$. If $t \to^*_{R,Ax} t'$ and t' is irreducible, we call t' *R,Ax-normal* and also say that t' is a *normal form* of t. We denote by $t\!\downarrow_{R,Ax}$ any normal form of t. Under suitable termination, confluence, and coherence conditions (see [19] for definitions), one can decompose an equational theory (Σ, E) into a rewrite theory (Σ, Ax, R) where $Ax \subseteq E$ and, for all terms $t, u \in \mathcal{T}_\Sigma(V)$, we have $t =_E u$ iff $t\!\downarrow_{R,Ax} =_{Ax} u\!\downarrow_{R,Ax}$. In this paper, we work with decomposable equational theories.

A rewriting theory R is *subterm-convergent* if it is convergent and, for each $l \to r \in R$, r is either a proper subterm of l or ground and in normal form with respect to R. For our soundness result, we consider the subclass $\mathcal{S}$ of subterm-convergent rewrite theories where each rule in R has one of the following forms.

- (R1): $d(c(x_1, \ldots, x_n, t), u) \to x_j$, where $c, d \in \Sigma_{\mathsf{pub}}$, t, u are terms, $j \in \widetilde{n}$, and $x_1, \ldots, x_n$ are pairwise distinct variables with $x_i \notin vars(t, u)$ for all $i \in \widetilde{n}$.
- (R2): $d(c(x_1, \ldots, x_n)) \to x_j$, where $c, d \in \Sigma_{\mathsf{pub}}$, $j \in \widetilde{n}$, and $x_1, \ldots, x_n$ are pairwise distinct variables.
- (R3): $c(x_1, \ldots, x_n) \to x_j$ where $c \in \Sigma_{\mathsf{pub}}$, x_j is a variable with $j \in \widetilde{n}$, and x_i is a variable or an atom for all $i \in \widetilde{n}$.
- (R4): $l \to a$ for a constant a.

Intuitively, the first three forms enable different types of projection of a term's arguments. Rules R1 and R2 apply a destructor d to extract one of c's arguments. In rule R1 the destructor has two arguments. The terms t and u can be seen a pair of matching keys required to extract x_j. Rule R3 uses no destructor. Finally, R4 models rewriting a term to a constant. Since the rules (R1-R3) have limited depth, we call the class $\mathcal{S}$ of rewrite theories *shallow subterm-convergent*.

We also introduce a condition on the equations Ax of the rewrite theory.

Definition 1. *A rewrite theory (Σ, Ax, R) is* well-formed *if for all $\{s, t\} \in Ax$, we have (i) neither s nor t is a pair and (ii) $top(s) = top(t)$.*

We only consider equational theories that can be decomposed into a shallow subterm-convergent, well-formed rewrite theory. These are adequate to model many well-known cryptographic primitives as illustrated by the examples below.

Example 1. We model the protocols of our case studies (see Sections 1 and 4) in the rewrite theory $\mathcal{R}_{cs} = (\Sigma_{cs}, Ax_{cs}, R_{cs})$ where

$$\Sigma_{cs} = \{\mathsf{sh}, \mathsf{pk}, \mathsf{pri}, \mathsf{prf}, \mathsf{kdf}, \mathsf{mac}, \langle\cdot,\cdot\rangle, \pi_1, \pi_2, \{\!|\cdot|\!\}_\cdot, \{\!|\cdot|\!\}_\cdot^{-1}, \{\cdot\}_\cdot, \{\cdot\}_\cdot^{-1}, [\cdot]_\cdot, \mathsf{ver}\} \cup \Sigma^0_{cs}$$

contains function symbols for: shared, public, and private long-term keys (where $\Sigma_{\mathsf{pri}} = \{\mathsf{sh}, \mathsf{pri}\}$); hash functions prf, kdf, and mac; pairs and projections; symmetric and asymmetric encryption and decryption; and signing and verification.

The set of atoms Σ^0_{cs} is specified later. The set R_{cs} consists of rewrite rules for projections (type R2) and for decryption and signature verification (type R1):

$$\pi_1(\langle t, u\rangle) \to t \qquad \{\lfloor\{\lfloor t\rfloor_k\rfloor_k^{-1} \to t \qquad \mathsf{ver}([t]_{\mathsf{pri}(k)}, \mathsf{pk}(k)) \to t$$
$$\pi_2(\langle t, u\rangle) \to u \qquad \{\{t\}_{\mathsf{pk}(k)}\}^{-1}_{\mathsf{pri}(k)} \to t$$

We have one equation in Ax_{cs}, namely, $\mathsf{exp}(\mathsf{exp}(g, X), Y) \simeq \mathsf{exp}(\mathsf{exp}(g, Y), X)$ to model Diffie-Hellman key exchange. We can also model Diffie-Hellman theories where the exponents constitute an abelian group.

Example 2. The theory of XOR is given by the following rewrite system where the rewrite rules are of type R3 and R4. The rightmost rule ensures coherence [19].

$$X \oplus Y \simeq Y \oplus X \qquad X \oplus 0 \to X \qquad X \oplus X \oplus Y \to Y$$
$$(X \oplus Y) \oplus Z \simeq X \oplus (Y \oplus Z) \qquad X \oplus X \to 0$$

For our theoretical development, we consider an arbitrary but fixed shallow subterm-convergent and well-formed rewrite theory (Σ, Ax, R) that includes the function symbols and rewrite rules for pairing and projections.

We denote by $dom(g)$ and $ran(g)$ the domain and range of a function g. We now define well-typed substitutions, which respect subtyping.

Definition 2 (Well-typed substitutions). *A substitution θ is* well-typed *if $\Gamma((X\theta)\downarrow_{R,Ax}) \preccurlyeq \Gamma(X)$ for all $X \in dom(\theta)$.*

2.3 Protocols

For a set of terms T, we define the set of events $Evt(T) = \{\mathsf{snd}(t), \mathsf{rcv}(t) \mid t \in T\}$ and $term(ev(t)) = t$ for event $ev(t)$. A *role* is a sequence of events from $Evt(\mathcal{M})$.

Definition 3 (Protocol). *A* protocol *is a function $P : \mathcal{V}_\alpha \rightharpoonup Evt(\mathcal{M})^*$ mapping agent variables to roles. Let $\mathcal{M}_P = term(ran(P))$ be the set of* protocol terms *appearing in the roles of P, and let $\mathcal{V}_P$, $\mathcal{A}_P$, $\mathcal{F}_P$, and $\mathcal{C}_P$ denote the sets of variables, agents, fresh values, and constants in $\mathcal{M}_P$.*

Example 3 ($\mathsf{IKE_m}$ protocol). We formalize the $\mathsf{IKE_m}$ protocol from Section 1 in the rewrite theory of Example 1 as follows, using upper-case (lower-case) identifiers for variables (atoms). The atoms Σ^0_{cs} are composed of constants $C = \{g, o, sA1, sA2, tSa, tSb\}$ and fresh values $F = \{na, nb, x, y, sPIa, sPIb\}$. The variables and their types are $A, B : \alpha$, $Ga, Gb : msg$, $SPIa, SPIb, Na, Nb : nonce$ where *nonce* is a user-defined type that satisfies $\beta_n \preccurlyeq_0 nonce$ for all $n \in F$. We show here the initiator role A. The responder role B is dual.

$$\begin{aligned}\mathsf{IKE_m}(A) = {} & \mathsf{snd}(sPIa, o, sA1, \mathsf{exp}(g, x), na) \cdot \mathsf{rcv}(sPIa, SPIb, sA1, Gb, Nb) \cdot \\ & \mathsf{snd}(sPIa, SPIb, \{\!|A, B, AUTHaa, sA2, tSa, tSb|\!\}_{SKa}) \cdot \\ & \mathsf{rcv}(sPIa, SPIb, \{\!|B, AUTHba, sA2, tSa, tSb|\!\}_{SKa})\end{aligned}$$

where the terms $SKa = \mathsf{kdf}(na, Nb, \mathsf{exp}(Gb, x), sPIa, SPIb)$ and

$$\begin{aligned}AUTHaa &= \mathsf{mac}(\mathsf{sh}(A, B), sPIa, o, sA1, \mathsf{exp}(g, x), na, Nb, \mathsf{prf}(SKa, A)) \\ AUTHba &= \mathsf{mac}(\mathsf{sh}(A, B), sPIa, SPIb, sA1, Gb, Nb, na, \mathsf{prf}(SKa, B)).\end{aligned}$$

represent the initiator A's view of the session key and of the authenticators.

$$\frac{u \in T}{T \vdash_E u}\ \mathsf{Ax} \qquad \frac{T \vdash_E t' \quad t' =_E t}{T \vdash_E t}\ \mathsf{Eq} \qquad \frac{T \vdash_E t_1 \quad \cdots \quad T \vdash_E t_n}{T \vdash_E f(t_1, \ldots, t_n)}\ \mathsf{Comp}\ (f \in \Sigma^{\geq 1}_{\mathsf{pub}})$$

Fig. 1. Intruder deduction rules (where $\Sigma^{\geq 1}_{\mathsf{pub}} = \Sigma^{\geq 1} \cap \Sigma_{\mathsf{pub}}$)

2.4 Operational semantics

Let *TID* be a countably infinite set of thread identifiers. When we instantiate a role into a thread for execution, we mark its variables and fresh values with the thread identifier i. We define the instantiation $t^{\#i}$ of a term t for $i \in \mathit{TID}$ as the term where every variable or fresh value u is replaced by u^i. Constants and agents remain unchanged. Instantiation does not affect the type of a term.

We define by $T^\sharp = \{t^{\#i} \mid t \in T \wedge i \in \mathit{TID}\}$ the set of instantiations of terms in a set T and abbreviate $T^\flat = T \cup T^\sharp$. For example, $\mathcal{M}^\sharp$ is the set of instantiated message terms, which we will use to instantiate roles into threads. We define the set of *network messages* exchanged during protocol execution by $\mathcal{N} = \mathcal{T}(\mathcal{V}^\sharp, \mathcal{A} \cup \mathcal{F}^\sharp \cup \mathcal{F}^\bullet \cup \mathcal{C})$, where $\mathcal{F}^\bullet = \{n^\bullet_k \mid n \in \mathcal{F} \wedge k \in \mathbb{N}\}$ is the set of attacker-generated fresh values. Note that $\mathcal{M}^\sharp \subseteq \mathcal{N}$. We abbreviate $\mathcal{T} = \mathcal{M} \cup \mathcal{N}$.

Semantics We use a Dolev-Yao attacker model parametrized by an equational theory E. Its judgements are of the form $T \vdash_E t$ meaning that the intruder can derive term t from the set of terms T. The derivable judgements are defined in a standard way by the three deduction rules in Figure 1.

We define a transition system with states (tr, th, σ), where

- tr is a *trace* consisting of a sequence of pairs of thread identifiers and events,
- $th : \mathit{TID} \rightharpoonup \mathit{dom}(P) \times \mathit{Evt}(\mathcal{M}^\sharp_P)^*$ are *threads* executing role instances, and
- $\sigma : \mathcal{V}^\sharp \rightharpoonup \mathcal{N}$ is a well-typed ground substitution from instantiated protocol variables to network messages such that $\mathcal{V}^\sharp_P \subseteq \mathit{dom}(\sigma)$.

The trace tr as well as the executing role instance are symbolic (with terms in $\mathcal{M}^\sharp$). The separate substitution σ instantiates these messages to (ground) network messages. The ground trace associated with such a state is $tr\sigma$.

The set Init_P of initial states of protocol P contains all (ϵ, th, σ) satisfying

$$\forall i \in \mathit{dom}(th).\ \exists R \in \mathit{dom}(P).\ th(i) = (R, P(R)^{\#i})$$

where all terms in the respective protocol roles are instantiated. The substitution σ is chosen non-deterministically in the initial state.

The rules in Figure 2 define the transitions. In both rules, the first premise states that a send or receive event heads thread i's role. This event is removed and added together with the thread identifier i to the trace tr. The substitution σ remains unchanged. The second premise of *RECV* requires that the network message $t\sigma$ matching the term t in the receive event is derivable from the intruder's

$$\frac{th(i) = (R, \mathsf{snd}(t) \cdot tl)}{(tr, th, \sigma) \to (tr \cdot (i, \mathsf{snd}(t)), th[i \mapsto (R, tl)], \sigma)}\ SEND$$

$$\frac{th(i) = (R, \mathsf{rcv}(t) \cdot tl) \quad IK(tr)\sigma \cup IK_0 \vdash_E t\sigma}{(tr, th, \sigma) \to (tr \cdot (i, \mathsf{rcv}(t)), th[i \mapsto (R, tl)], \sigma)}\ RECV$$

Fig. 2. Operational semantics

(ground) knowledge $IK(tr)\sigma \cup IK_0$. Here, $IK(tr)$ denotes the (symbolic) intruder knowledge derived from a trace tr as the set of terms in the send events on tr, i.e., $IK(tr) = \{t \mid \exists i.\ (i, \mathsf{snd}(t)) \in tr\}$ and IK_0 denotes the intruder's (ground) initial knowledge. We assume $\mathcal{A} \cup \mathcal{C} \cup \mathcal{F}^{\bullet} \subseteq IK_0$ and IK_0 is R,Ax-normal. Note that the *SEND* rule implicitly updates this intruder knowledge.

2.5 Property language

We use the same first-order specification language as in [23] to express secrecy and authentication properties. Therefore, we only sketch some of its elements and give examples. There are atomic formulas to express equality ($t = u$), the secrecy of a term ($secret(t)$), the occurrence of an event e by thread i in the trace ($steps(i, e)$), the fact that thread i executes role R, and the honesty of agents instantiating role R in the view of thread i. Quantification is allowed over thread identifier variables. To achieve attack preservation, we focus on the fragment of this logic where the predicate $secret(t)$ occurs only positively.

Example 4 (Properties of $\mathsf{IKE_m}$*).* We express the secrecy of the Diffie-Hellman key $\mathsf{exp}(Gb, x)$ for role A of the protocol $\mathsf{IKE_m}$ of Example 3 as follows.

$$\phi_s = \forall j.\ (role(j, A) \wedge honest(j, [A, B]) \wedge steps(j, \mathsf{rcv}(t_4))) \Rightarrow secret(\mathsf{exp}(Gb^j, x^j)).$$

where $t_4 = \langle sPIa, SPIb, \{\!|B, AUTHba, sA2, tSa, tSb|\!\}_{SKa}\rangle$ and $honest(j, [A, B])$ means that A and B are honest. We formalize non-injective agreement of A with B [21] on the nonces na and nb and the DH half-keys $\mathsf{exp}(g, x)$ and $\mathsf{exp}(g, y)$ by

$$\begin{aligned}\phi_a = \forall j.\ & (role(j, A) \wedge honest(j, [A, B]) \wedge steps(j, \mathsf{rcv}(t_4))) \\ & \Rightarrow (\exists k.\ role(k, B) \wedge steps(k, \mathsf{snd}(\langle SPIa, sPIb, sA1, \mathsf{exp}(g, y), nb\rangle)) \wedge \\ & \quad \langle A^j, B^j, na^j, Nb^j, \mathsf{exp}(g, x^j), Gb^j\rangle = \langle A^k, B^k, Na^k, nb^k, Ga^k, \mathsf{exp}(g, y^k)\rangle).\end{aligned}$$

3 Security protocols abstractions

We introduce our security protocol abstractions and illustrate their usefulness on our running example. We will present two types of protocol abstractions:

Typed abstractions transform a term's structure by reordering or removing fields and by splitting or removing cryptographic operations. The same transformations are applied to all terms of a given type and its subtypes.

Untyped abstractions complement typed ones with additional simplifications: the removal of unprotected atoms and variables and of redundant subterms.

Our main results are soundness theorems for these abstractions. They ensure that any attack on a given property of the original protocol translates to an attack on a related property in the abstracted protocol. As we will explain, these results only hold under certain conditions on the protocol and the property. Here, we will mainly focus on typed abstractions, but we will also briefly introduce the untyped ones (for more details, see Appendices G.1 to G.3).

3.1 Typed protocol abstractions

Our typed abstractions are specified by a list of recursive equations subject to some conditions on their shape. We define their semantics in terms of a simple Haskell-style functional program. We use both pattern matching on terms and subtyping on types to select the equation to be applied to a given term. This ensures that terms of related types are transformed in a uniform manner.

Syntax Let $\mathcal{W} = \{\mathcal{W}_\tau\}_{\tau \in \mathcal{Y}}$ be a family of *pattern variables* disjoint from $\mathcal{V}$. We define the set of *patterns* by $\mathcal{P} = \mathcal{T}(\mathcal{W}, \emptyset)$. A pattern $p \in \mathcal{P}$ is called *linear* if each (pattern) variable occurs at most once in p. We extend the typing function Γ to patterns by setting $\Gamma(X) = \tau$ iff $X \in \mathcal{W}_\tau$ and then lifting it homomorphically to all patterns. Our typed message abstractions are instances of the following recursive function specifications.

Definition 4. *A* function specification $F_f = (f, E_f)$ *consists of an unary function symbol* $f \notin \Sigma^1$ *and a list of equations*

$$E_f = [f(p_1) = u_1, \ldots, f(p_n) = u_n],$$

where each $p_i \in \mathcal{P}$ *is a linear pattern such that* $u_i \in \mathcal{T}_{\Sigma^{\geq 1} \cup \{f\}}(vars(p_i))$ *for all* $i \in \widetilde{n}$*, i.e.,* u_i *consists of variables from* p_i *and function symbols from* $\Sigma^{\geq 1} \cup \{f\}$*.*

Given a list of equations L, we denote the set of its pattern types by $\Pi(L) = \{\Gamma(p) \mid (f(p) = u) \in L\}$ and define $\Pi_f = \Pi(E_f)$.

To ensure the *uniform* abstraction of terms of a type τ and its subtypes (e.g., a term t and its well-typed instance $t\theta$), we require pairwise disjoint patterns.

Definition 5. *A function specification* $F_f = (f, E_f)$ *is* pattern-disjoint *if the types in* Π_f *are pairwise disjoint, i.e., for all* $i, j \in \widetilde{n}$ *such that* $i \neq j$*, we have* $\Gamma(p_i)\downarrow \cap \Gamma(p_j)\downarrow = \emptyset$*.*

We use vectors (lists) of terms $\bar{t} = [t_1, \ldots, t_n]$ for $n > 0$. We define $set(\bar{t}) = \{t_1, \ldots, t_n\}$ and $\widehat{f}(\bar{t}) = \langle f(t_1), \ldots, f(t_n)\rangle$, the elementwise application of a function f to a vector where the result is converted to a tuple (with the convention

$\langle t \rangle = t$). We extend *split* to vectors by $split(\overline{t}) = split(set(\overline{t}))$. We define three sets of function symbols occurring in R and Ax as follows.

$$\begin{aligned}
\mathsf{C_R} &= \{c \mid d(c(x_1, \ldots, x_n, t), u) \to x_j \in R\} \\
\mathsf{C_{Key}} &= \bigcup\{ct(t) \cup ct(u) \mid d(c(x_1, \ldots, x_n, t), u) \to x_j \in R\} \\
\mathsf{C_{Ax}} &= \bigcup\{ct(s) \cup ct(t) \mid \{s, t\} \in Ax\}
\end{aligned}$$

The function $pp(c)$ returns the set of extractable indices of a function symbol c, i.e., $pp(c) = \{j \mid d(c(x_1, \ldots, x_n, t), u) \to x_j \in R \text{ or } d(c(x_1, \ldots, x_n)) \to x_j \in R\}$.

Definition 6 (Typed abstraction). *A pattern-disjoint function specification $F_f = (f, E_f)$ is a* typed abstraction *if each equation in E_f has the form*

$$f(c(p_1, \ldots, p_n)) = \langle e_1, \ldots, e_d \rangle$$

where for each $i \in \widetilde{d}$ we have either

(a) $e_i = f(q)$ such that $q \in split(p_j)$ for some $j \in \widetilde{n}$, or

(b) $e_i = c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n}))$ such that $set(\overline{q_j}) \subseteq split(p_j)$ for all $j \in \widetilde{n}$, $c \neq \langle \cdot, \cdot \rangle$, and $c \in \mathsf{C_R}$ implies $\overline{q_n} = [p_n]$, i.e., $\widehat{f}(\overline{q_n}) = f(p_n)$.

Moreover, we require

(i) for all $j \in pp(c)$ we have $split(p_j) \subseteq Q_j$ where

$$Q_j = \bigcup\{set(\overline{q_j}) \mid \exists i.\, e_i = c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n}))\} \cup \{q \mid \exists i.\, e_i = f(q)\}.$$

(ii) if $c \in \mathsf{C_{Ax}} \cup \mathsf{C_{Key}}$ then $p_i = X_i : msg$ for all $i \in \widetilde{n}$, $d = 1$ and $e_1 = c(f(X_1), \ldots, f(X_n))$ is an instance of (b). We say f is homomorphic on c.

Intuitively, the abstractions can only weaken the cryptographic protection of terms, but never strengthen it. Each defining equation maps a term with top-level symbol c to a tuple whose components have the form (a) or (b). Form (a) allows us to pull fields out of the scope of c, hence removing c's protection. Using form (b) we can reorder or remove fields in each argument of c. Form (b) is subject to two conditions. First, we disallow this form for pairs to obtain the simple shape $f(\langle p_1, p_2 \rangle) = \widehat{f}(\overline{q})$. Second, we cannot permit the reordering or removal of fields in key positions, i.e., in the last argument of $c \in \mathsf{C_R}$. Moreover, by point (i), all fields of extractable arguments, i.e., elements of $split(p_j)$ for $j \in pp(c)$, must be present in some e_i and point (ii) requires that the abstraction is homomorphic for function symbols c occurring in axioms and in keys ($c \in \mathsf{C_{Ax}} \cup \mathsf{C_{Key}}$).

Example 5. Let $F_f = (f, E_f)$ be the typed abstraction and $X : \gamma_c$, $Y : nonce$, and $Z, U, V : msg$ where E_f consists of the following equations:

$$\begin{aligned}
f(\langle X, Y, Z \rangle) &= \langle f(Y), f(X), f(Z) \rangle \\
f(\mathsf{kdf}(X, Y, U, V)) &= \langle \mathsf{kdf}(f(X), f(Y)), \mathsf{kdf}(f(U)) \rangle \\
f(\{\!|X, Y, Z|\!\}_U) &= \langle \{\!|f(X), f(Y)|\!\}_{f(U)}, f(Y), \{\!|f(Z)|\!\}_{f(U)} \rangle
\end{aligned}$$

Note that X and Y only match the constant c and a nonce, respectively. Clearly, F_f is pattern-disjoint. The first clause swaps the first two fields in n-tuples for $n \geq 3$. The second one splits a kdf hash into two, dropping the field V. The last clause splits an encryption: the pair $\langle f(X), f(Y) \rangle$ and $f(Z)$ are encrypted separately with the key $f(U)$ and $f(Y)$ is pulled out of the encryption.

$$\textbf{fun } f(t) = \textbf{case } t \textbf{ of}$$
$$\quad \|_{(f(p)=u)\in E_f^+} \;\; p \mid \Gamma(t) \preccurlyeq \Gamma(p) \Rightarrow u$$

Program 1. Functional program f resulting from $F_f = (f, E_f)$.

Semantics The semantics of a typed abstraction F_f is given by the Haskell-style functional program f (Program 1).[1] To ensure totality, we use the extended function specification $(f, E_f^+) = (f, E_f \cdot E_f^0)$, where $f(g(Z_1, \ldots, Z_n)) = g(f(Z_1), \ldots, f(Z_n)) \in E_f^0$ for each $g \in \Sigma^n$ with $n \geq 1$ such that $Z_i \colon msg$ for all $i \in \widetilde{n}$, and $f(Z) = Z$ with $Z \colon msg$ is the last clause in E_f^0. We assume E_f and E_f^0 do not share variables. The **case** statement has a clause

$$p \mid \Gamma(t) \preccurlyeq \Gamma(p) \Rightarrow u$$

for each equation $f(p) = u$ of E_f^+. Such a clause is enabled if (1) the term t matches the pattern p, i.e., $t = p\theta$ for some substitution θ, and (2) its type $\Gamma(t)$ is a subtype of $\Gamma(p)$. The first enabled clause is executed. Hence, the equations E_f^0 serve as fall-back clauses, which cover the terms not handled by E_f.

We extend f to events, event sequences, and traces by applying f to the terms they contain and to substitutions and protocols by applying f to the terms in their range. Similarly, we extend f to formulas ϕ of our property language by applying f to all terms occurring in ϕ.

Finding abstractions Finding abstractions is fully automated by our tool using a heuristic (see Section 4). However, the resulting abstractions can be counterintuitive. Therefore, we present here a simple strategy that we apply to our running example below. We start by identifying the terms that appear in the $secret(\cdot)$ predicates and equations of the desired properties. Then we determine the cryptographic operations that are essential to achieve these properties and try to remove all other terms and operations.

Example 6 (from $\mathsf{IKE_m}$ *to* $\mathsf{IKE_m^1}$*).* In order to preserve the secrecy of the DH key $\mathsf{exp}(\mathsf{exp}(g, x), y)$ and the agreement on na, nb, $\mathsf{exp}(g, x)$, and $\mathsf{exp}(g, y)$, we have to keep either the mac or the symmetric encryption with SK (see Examples 3 and 4). We want to remove as many other fields and operations as possible (e.g., prf). We choose to remove the encryption as this allows us to later remove additional fields (e.g., $sA2$) using untyped abstractions. We keep o in $AUTHa$ to prevent unifiability with $AUTHb$ and hence potential false negatives. This leads us to the typed abstraction $F_1 = (f_1, E_1)$ where E_1 is defined by the equations

[1] We are overloading the symbol f here, but no confusion should arise.

(omitting the homomorphic clauses for exp and $\langle\cdot,\cdot\rangle$)

$$\begin{aligned} f_1(\{\!|X,Y|\!\}_Z) &= \langle f_1(X), f_1(Y)\rangle \\ f_1(\mathsf{mac}(X_1,\ldots,X_8)) &= \mathsf{mac}(\widehat{f_1}([X_1,X_3,X_5,X_6,X_7,X_8])) \\ f_1(\mathsf{mac}(Y_1,\ldots,Y_8)) &= \mathsf{mac}(\widehat{f_1}([Y_1,Y_5,Y_6,Y_7,Y_8])) \\ f_1(\mathsf{kdf}(Z_1,\ldots,Z_5)) &= \mathsf{kdf}(f_1(Z_3)) \\ f_1(\mathsf{prf}(U,Z)) &= f_1(U) \end{aligned}$$

and $X\colon\alpha$, $X_3\colon\gamma_o$, $Y_3\colon \mathit{nonce}$, $Z_3\colon \mathsf{exp}(\mathit{msg},\mathit{msg})$, $U\colon \mathsf{kdf}(\mathit{msg})$ and all remaining pattern variables are of type msg. Clearly, F_1 is pattern-disjoint. Applying f_1 to IKE_m we obtain $\mathsf{IKE}^1_\mathsf{m}$. Here is the abstracted initiator role.

$$\begin{aligned} S_{\mathsf{IKE}^1_\mathsf{m}}(A) = \mathsf{snd}(&\mathit{sPIa}, o, \mathit{sA1}, \mathsf{exp}(g,x), \mathit{na})\cdot \mathsf{rcv}(\mathit{sPIa}, \mathit{SPIb}, \mathit{sA1}, \mathit{Gb}, \mathit{Nb})\cdot \\ &\mathsf{snd}(\mathit{sPIa}, \mathit{SPIb}, A, B, \mathit{AUTHaa}, \mathit{sA2}, \mathit{tSa}, \mathit{tSb})\cdot \\ &\mathsf{rcv}(\mathit{sPIa}, \mathit{SPIb}, B, \mathit{AUTHba}, \mathit{sA2}, \mathit{tSa}, \mathit{tSb}) \end{aligned}$$

with $\mathit{SKa} = \mathsf{kdf}(\mathsf{exp}(\mathit{Gb},x))$, $\mathit{AUTHaa} = \mathsf{mac}(\mathsf{sh}(A,B), o, \mathsf{exp}(g,x), \mathit{na}, \mathit{Nb}, \mathit{SKa})$, and $\mathit{AUTHba} = \mathsf{mac}(\mathsf{sh}(A,B), \mathit{Gb}, \mathit{Nb}, \mathit{na}, \mathit{SKa})$. In a second step, we will remove most fields in the roles of $\mathsf{IKE}^1_\mathsf{m}$ using untyped abstractions.

3.2 Soundness of typed abstractions

To justify the soundness of our abstractions, we show that any attack on a property ϕ of the original protocol P is reflected as an attack on the property $f(\phi)$ of the abstracted protocol $f(P)$. We decompose this into reachability preservation (RP) and an attack preservation (AP) as follows. We require that, for all reachable states (tr, th, σ) of P, there is a ground substitution σ' such that

(RP) $(f(tr), f(th), \sigma')$ is a reachable state of $f(P)$, and
(AP) $(tr, th, \sigma) \not\models \phi$ implies $(f(tr), f(th), \sigma') \not\models f(\phi)$.

These properties will require some assumptions about the protocol P, the formula ϕ, and the abstraction f. Before we formally state the soundness theorem, we will introduce and motivate these assumptions while sketching its proof. For the remainder of this subsection we assume arbitrary but fixed P, ϕ, F_f.

We first define the *uniform domain* of F_f as the set of terms that it transforms using only the clauses of E_f, without recourse to the fall-back clauses of E^0_f.

Definition 7 (Uniform domain). *Let $\mathit{Rec}(F_f, t)$ be the set of terms u such that $f(u)$ is called in the computation of $f(t)$. We define the* uniform domain *of F_f by $\mathit{udom}(F_f) = \{t \in \mathcal{T} \mid \Gamma(\mathit{Rec}(F_f,t)) \subseteq \Pi_f{\downarrow} \cup \mathcal{Y}_{at}\}\}$.*

We will require that the protocol terms $t \in \mathcal{M}_P$ belong to this set, which ensures that their instances $t\theta$ with R,Ax-normal substitutions θ are transformed uniformly. This is captured and slightly extended in the following theorem.

Theorem 1 (Substitution property). *Let θ be a substitution and $t \in \mathcal{T}$. If either (i) f is homomorphic for $\mathit{ct}(t)$, or (ii) $t \in \mathit{udom}(F_f)$ and θ is well-typed and R,Ax-normal, then $f(t\theta) = f(t)f(\theta)$.*

A second natural ingredient that we need in our soundness proof is that abstractions preserve equality modulo E. This does however not hold in general. The following condition ensures that F_f is well-behaved with respect to the axioms Ax and rewrite rules R.

Definition 8 (R, Ax-closedness). *F_f is R, Ax-*closed *if the following holds.*

(i) $t =_{Ax} u$ *implies* $\tau_t \preccurlyeq \tau$ *iff* $\tau_u \preccurlyeq \tau$*, for all R,Ax-normal composed ground terms* $t : \tau_t$ *and* $u : \tau_u$ *and all* $\tau \in \Pi(E_f^+)$*, and*
(ii) $s, t, l \in \mathit{udom}(F_f)$ *for all equations* $s \simeq t \in Ax$ *and all rules* $l \to r \in R$*.*

In Appendix F, we present a syntactic criterion to check point (i). We henceforth assume that F_f is R, Ax-closed. We must also restrict the interference of protocol terms with the rewrite rules. We could therefore require these terms to be R,Ax*-stable*, i.e., R,Ax-normal substitutions do not create redexes. However, this would exclude many protocols such as those using XOR. Here is a weaker condition.

Definition 9 (R, Ax-homomorphism). *We say that f is R, Ax-*homomorphic *for a term t if for all non-variable positions p in t and for all rules $l \to r \in R$ such that there exists an Ax-unifier of $t|_p$ and l that is well-typed, it holds that*

(i) f is homomorphic for all $c \in ct(l)$,
(ii) f is homomorphic for $top(t|_{p'})$ for all strict prefixes p' of p.

We define $\mathit{rdom}(F_f)$ to be the set of terms for which f is R, Ax-homomorphic.

For example, the term $t = X \oplus k$ is not R,Ax-stable, but any F_f that is homomorphic for $\oplus$ is R, Ax-homomorphic for t. Many interesting protocols P satisfy $\mathcal{M}_P \subseteq \mathit{rdom}(F_f)$, including those from our case studies. We establish equality preservation for terms in $\mathit{rdom}(F_f)$.

Theorem 2 (Equality preservation). *Let t and u be terms such that $t, u \in \mathit{rdom}(F_f)$. Then $t =_E u$ implies $f(t) =_E f(u)$.*

Reachability preservation (RP) To achieve reachability preservation, we prove that every step of P can be simulated by a corresponding step of $f(P)$. In particular, to simulate receive events, we show that intruder deducibility is preserved under abstractions f (cf. second premise of rule $RECV$), i.e.,

$$T\theta, IK_0 \vdash_E u\theta \;\Rightarrow\; f(T)f(\theta{\downarrow_{R,Ax}}), f(IK_0) \vdash_E f(u)f(\theta{\downarrow_{R,Ax}}). \tag{1}$$

This property is also required to show the preservation of attacks on secrecy as part of (AP). We first establish deducibility preservation for ground terms:

Theorem 3 (Deducibility preservation). *Let $T \cup \{t\} \subseteq \mathcal{N}$ be a set of ground network messages such that $\mathcal{C} \subseteq T$ and T is R,Ax-normal. Then $T \vdash_E t$ implies $f(T) \vdash_E f(t{\downarrow_{R,Ax}})$.*

We can now derive (1) by applying Theorems 3, 2 and 1 in this order, combined with applications of rule Eq and a cut property of intruder deduction. Summarizing, reachability preservation (RP) holds for $\mathcal{M}_P \subseteq \mathit{udom}(F_f) \cap \mathit{rdom}(F_f)$.

Attack preservation (AP) We next define and explain the conditions on formulas needed to establish attack preservation. Let

- Sec_ϕ be the set of all terms t that occur in formulas $secret(t)$ in ϕ,
- Eq_ϕ be the set of pairs (t, u) such that the equation $t = u$ occurs in ϕ and $EqTerm_\phi = \{t, u \mid (t, u) \in Eq_\phi\}$ is the set of underlying terms, and
- Evt_ϕ be the set of events occuring in ϕ.

Let Eq_ϕ^+ the positively occurring equations in ϕ and similarly for Evt_ϕ.

Definition 10 (Safe formulas). *ϕ is* safe *for P and f if*

(i) $Sec_\phi \cup EqTerm_\phi \subseteq udom(F_f) \cap rdom(F_f)$,
(ii) $f(t\sigma) =_E f(u\sigma)$ *implies* $t\sigma =_E u\sigma$ *for all* $(t, u) \in Eq_\phi^+$ *and for all well-typed and R,Ax-normal ground substitutions σ, and*
(iii) $f(t) = f(u)$ *implies* $t = u$, *for all* $e(t) \in Evt_\phi^+$ *and* $e(u) \in Evt(\mathcal{M}_P)$.

Condition (i) requires that F_f is uniform and R, Ax-homomorphic for the terms in secrecy statements and equalities. Condition (ii) is required to preserve attacks on agreement properties. Concretely, it expresses injectivity of the abstraction on the terms in positively occurring equalities. In Appendix F, we provide a syntactic criterion to check condition (ii) that avoids the universal quantification over substitutions. Condition (iii) is required for properties involving event orderings and *steps* predicates. It states that the abstraction must not identify an event occurring positively in the property with a distinct protocol event.

We now state the soundness theorem. Below, IK_0 and IK_0' respectively denote the intruder's initial knowledge associated with P and $f(P)$.

Theorem 4 (Soundness). *Suppose P, ϕ, and F_f satisfy (i) $f(IK_0) \subseteq IK_0'$, (ii) F_f is R, Ax-closed, (iii) $\mathcal{M}_P \subseteq udom(F_f) \cap rdom(F_f)$, and ϕ is safe for P and f. Then, for all states (tr, th, σ) reachable in P, we have*

1. *$(f(tr), f(th), f(\sigma{\downarrow_{R,Ax}}))$ is a reachable state of $f(P)$, and*
2. *$(tr, th, \sigma) \not\models \phi$ implies $(f(tr), f(th), f(\sigma{\downarrow_{R,Ax}})) \not\models f(\phi)$.*

3.3 Untyped abstractions

Typed abstractions offer a wide range of possibilities to transform cryptographic operations including subterm removal, splitting, and pulling fields outside a cryptographic operation. We complement these abstractions with two kinds of *untyped abstractions* that allow us to remove (1) unprotected atoms and variables of any type and (2) redundancy in the form of intruder-derivable terms. Untyped protocol abstractions are functions $g : \mathcal{T} \to \mathcal{T} \cup \{\mathsf{nil}\}$ where messages to be removed are mapped to nil. We remove events with nil arguments from the roles. Here we have to content ourselves with an example and refer the reader to Appendix G for the details and soundness results.

Atom/variable removal The *removal abstraction* $rem_T : \mathcal{T} \to \mathcal{T} \cup \{\mathsf{nil}\}$ for a set T of atoms or variables is defined by

- $rem_T(u) = \mathsf{nil}$ if $u \in T^\flat$,
- $rem_T(\langle t_1, t_2 \rangle) = \begin{cases} rem_T(t_i) & \text{if } rem_T(t_{3-i}) = \mathsf{nil} \text{ for } i \in \widetilde{2} \\ \langle rem_T(t_1), rem_T(t_2) \rangle & \text{otherwise} \end{cases}$
- $rem_T(t) = t$ for all other terms.

In order to preserve attacks, we have to restrict the removal of atoms and variables from a protocol term t to fields $u \in \mathit{split}(t)$ that appear only unprotected (clear) in t, i.e., such that $u \notin \mathit{subs}(t) \setminus \mathit{split}(t)$.

Example 7 ($\mathsf{IKE}^1_\mathsf{m}$ *to* $\mathsf{IKE}^2_\mathsf{m}$*).* We use atom/variable removal to simplify the protocol $\mathsf{IKE}^1_\mathsf{m}$. First, we recall the specification of role A of $\mathsf{IKE}^1_\mathsf{m}$.

$$\begin{aligned} S_{\mathsf{IKE}^1_\mathsf{m}}(A) = {} & \mathsf{snd}(\mathit{sPIa}, o, \mathit{sA1}, \mathsf{exp}(g, x), na) \cdot \mathsf{rcv}(\mathit{sPIa}, \mathit{SPIb}, \mathit{sA1}, \mathit{Gb}, \mathit{Nb}) \cdot \\ & \mathsf{snd}(\mathit{sPIa}, \mathit{SPIb}, A, B, \mathit{AUTHaa}, \mathit{sA2}, \mathit{tSa}, \mathit{tSb}) \cdot \\ & \mathsf{rcv}(\mathit{sPIa}, \mathit{SPIb}, B, \mathit{AUTHba}, \mathit{sA2}, \mathit{tSa}, \mathit{tSb}) \end{aligned}$$

We remove the role names A and B, the constants o, $\mathit{sA1}$, $\mathit{sA2}$, tSa, tSb, the fresh value sPIa, and the variable SPIb using an atom/variable removal abstraction. The result is the protocol $\mathsf{IKE}^2_\mathsf{m}$ whose initiator role is defined as follows.

$$S_{\mathsf{IKE}^2_\mathsf{m}}(A) = \mathsf{snd}(\mathsf{exp}(g, x), na) \cdot \mathsf{rcv}(\mathit{Gb}, \mathit{Nb}) \cdot \mathsf{snd}(\mathit{AUTHaa}) \cdot \mathsf{rcv}(\mathit{AUTHba})$$

We also apply the typed abstraction from Example 6 and the untyped abstraction here to the properties ϕ_s and ϕ_a of Example 4. These only affect the events in the *steps* predicates. The relevant soundness conditions are satisfied.

Redundancy removal A redundancy removal abstraction rd enables the elimination of redundancies within each role of a protocol. Intuitively, a protocol term t appearing in a role r can be abstracted to $rd(t)$ if t and $rd(t)$ are derivable from each other under the intruder knowledge T containing the terms preceding t in r and the initial knowledge IK_0. For example, we can simplify $r = \pi_2(t) \cdot \mathsf{rcv}(\langle t, u \rangle)$ to $\pi_2(t) \cdot \mathsf{rcv}(u)$. In contrast to atom/variable removal, redundancy removal can also remove composed terms. It is therefore a very effective ingredient for automatic abstraction, which we describe next.

4 Implementation and experimental results

We have implemented our abstraction methodology for the Scyther tool and tested it on a variety of complex protocols, mainly from the IKE and ISO/IEC 9798 families. Scyther is an efficient verifier for security protocols. It supports verification for both a bounded and an unbounded number of threads. Protocols are specified by a set of linear role scripts. It also supports user-defined types. These features match our setting very well.

4.1 Abstraction module for the Scyther tool

To simplify a protocol, the abstraction module first applies a typed abstraction and then simplifies the resulting protocol using a redundancy abstraction, followed by an atom/variable removal abstraction. Typed abstractions are generated automatically, guided by a heuristic. However, users have the possibility to provide (possibly partial) specifications of typed abstractions.

Central to our heuristic is the notion of *essential terms*, i.e., long-term keys or (sub)terms of Sec_ϕ and $EqTerm_\phi$ for a given property ϕ. The heuristic assigns *security labels*, c for confidentiality and a for authenticity, to cryptographic primitives as their intended security guarantees. These labels are inherited by subterms. In particular, we label symmetric-key encryptions and MACs with c and a, asymmetric encryptions and hashes with c, and signatures with a. Based on this labeling, we decide which terms are removed, pulled out, or kept. The main criterion is that such transformations must preserve the following labeling properties of each essential term t: the presence of an a label on *some* occurrence of t and of c labels on *all* occurrences of t. The heuristic may compute successively more abstract versions of a protocol, proceeding from the outside to the inside of terms in several rounds.

Example 8. We can simplify the term $\{\!|B, AUTHba, sA2, tSa, tSb|\!\}_{SKa}$ where $AUTHba = \mathsf{mac}(\underline{\mathsf{sh}(A,B)}, sPIa, SPIb, sA1, \underline{Gb}, \underline{Nb}, \underline{na}, \mathsf{prf}(SKa, B))$ of the $\mathsf{IKE_m}$ protocol from Example 3 in two successive abstraction steps as follows.

$$\begin{aligned}\{\!|B, AUTHba, sA2, tSa, tSb|\!\}_{SKa} &\mapsto \langle B, AUTHba, sA2, tSa, tSb\rangle\\ AUTHba &\mapsto \mathsf{mac}(\underline{\mathsf{sh}(A,B)}, \underline{Gb}, \underline{Nb}, \underline{na}, \mathsf{prf}(SKa, B))\end{aligned}$$

In the first step, we pull the whole plaintext out of the encryption since the security labels of essential terms (underlined) are preserved by the mac. In the second step, we transform $AUTHba$ by keeping essential and removing inessential terms. Note that removing the term $u = \mathsf{prf}(SKa, B)$ or pulling it out of the mac would not preserve authenticity for the essential term x inside SKa. In a further step, we can simplify u by deleting inessential subterms and dropping prf.

Our abstractions are sound, but not complete. Therefore, we may encounter false negatives. We carefully try to avoid the introduction of such spurious attacks, e.g., by checking that abstractions do not introduce new pairs of unifiable terms. We currently do not check automatically whether an attack is spurious. Whenever an attack on a protocol P is found, we proceed to analyze (only) the failed properties on the next more concrete protocol in the series of abstractions.

4.2 Experimental results

We have validated the effectiveness of our abstractions on 22 members of the IKE and ISO/IEC 9798 protocol families and on the PANA-AKA protocol [3]. We verify these protocols using five tools based on four different techniques: Scyther [10], CL-Atse [27], OFMC [5], SATMC [4], and ProVerif [6]. Only Scyther

protocol	No	S	A	W	N	3	4	5	6	7	8	∞
IKEv1-pk2-a2	1	✓			✓	44.01	302.86	1843.80	10999.80	TO	TO	TO
						6.01	26.36	151.88	1014.63	6838.97	TO	TO
IKEv1-pk-a22	1	✓			✓	18.48	82.93	249.55	554.09	1006.04	1734.85	TO
						0.83	1.26	2.08	3.47	5.96	10.28	TO
IKEv2-eap	5	✓			✓	TO	TO	TO	TO	TO	TO	TO
						78.35	798.44	4212.71	20911.20	TO	TO	TO
IKEv2-mac	5	✓			✓	1.85	4.91	6.72	8.07	8.42	8.49	8.70
						0.62	1.77	1.83	1.73	1.73	1.80	1.74
IKEv2-mactosig	4	✓			✓	11.65	141.37	1075.46	7440.81	TO	TO	TO
						2.89	12.38	24.54	38.68	53.36	65.07	77.68
IKEv2-sigtomac	5	✓			✓	6.15	33.19	65.05	115.34	204.93	206.45	237.34
						3.59	12.72	28.44	44.44	55.11	66.97	67.15
IKEv1-pk-m	2				×	48.62	269.92	507.40	869.23	16254.80	TO	TO
						0.04	0.05	0.05	0.05	0.05	0.05	TO
IKEv1-pk-m2	2				✓/×	18.26	274.87	4438.72	TO	TO	TO	TO
						1.48	7.79	32.75	110.32	339.93	963.08	TO
IKEv1-sig-m	2				×	0.34	0.45	0.45	0.45	0.45	0.46	0.44
						0.05	0.05	0.05	0.06	0.05	0.05	0.06
IKEv1-sig-m-perlman	2				×	2.86	13.99	40.78	67.83	72.08	72.15	109.03
						0.05	0.05	0.05	0.05	0.05	0.05	0.05
ISO/IEC 9798-2-5	1	✓				0.78	8.96	73.87	564.67	4214.22	TO	TO
						0.07	0.11	0.12	0.11	0.11	0.11	0.11
ISO/IEC 9798-2-6	1	✓				0.57	3.74	18.42	67.01	196.30	488.04	21278.58
						0.05	0.04	0.05	0.05	0.05	0.05	0.05
ISO/IEC 9798-3-6-1	2		✓		✓	43.08	802.95	8903.70	ME	ME	ME	ME
						0.13	0.18	0.19	0.19	0.19	0.19	0.19
ISO/IEC 9798-3-6-2	1		✓		✓	2.74	8.67	19.56	33.91	52.51	69.48	90.04
						0.12	0.15	0.15	0.15	0.15	0.15	0.15
ISO/IEC 9798-3-7-1	2		✓		✓	40.43	740.47	7483.36	16631.42	ME	ME	ME
						0.13	0.18	0.19	0.19	0.19	0.19	0.19
ISO/IEC 9798-3-7-2	1		✓		✓	2.38	7.71	16.68	26.99	35.06	49.49	TO
						0.22	0.32	0.33	0.33	0.33	0.33	0.33
PANA-AKA	5	✓	✓	✓	✓	5769.53	TO	TO	TO	TO	TO	TO
						0.10	0.10	0.10	0.10	0.10	0.10	0.10

Table 1. Experimental results. The time is in seconds. **No**: Number of abstractions. Properties: **S**ecrecy, **A**liveness, **W**eak agreement, and **N**on-injective agreement.

and ProVerif support verification of an unbounded number of threads. In Table 1, we present here a selection of the experimental results for Scyther and refer to Appendix H for a complete account, including results for other tools. Our models of the IKE and ISO/IEC 9798 protocols are based on Cremers' [8,9]. Since Scyther uses a fixed signature with standard cryptographic primitives and no equational theories, the IKE models approximate the DH theory by oracle roles.

We mark verified properties by ✓ and falsified ones by ×. An entry ✓/× means the property holds for one role but not for the other. Each row consists of two lines, corresponding to the analysis time without (line 1) and with (line 2) abstraction for 3-8 or unboundedly many (∞) threads. We ran the experiments on a cluster of 12-core AMD Opteron 6174 processors each with 64 GB RAM.

Verification For 9 of the 13 original protocols that are verified, an unbounded verification attempt results in a timeout (TO, set to 8h cpu time) or memory exhaustion (ME). In 5 of these cases our abstractions enabled a verification in less than 0.4 seconds and in one case in 78 seconds. However, for the first three protocols, we still get a timeout for the abstractions. For the large majority of the bounded verification tasks, we significantly push the bound on the number of threads and achieve massive speedups. For example, our abstractions enable the verification of the complex nested protocols IKEv2-eap and PANA-AKA. Scyther verifies an abstraction of IKEv2-eap for up to 6 threads where it times out on the original even for 3 threads. More strikingly, it completes an unbounded verification of the simplified PANA-AKA in less than 1/10 second whereas it failed on the original already for 4 threads. We also achieve dramatic speedups for many other protocols, most notably for the IKEv1-pk-a22, ISO/IEC 9798-2-6, and ISO/IEC 9798-3-6-2 protocols. Moreover, the verification time for many abstracted protocols increases much more slowly than for their originals. For instance, we obtain almost constant verification times for the six ISO/IEC 9798 protocols, whereas the time significantly increases on some originals, e.g., for ISO/IEC 9798-3-6-1. However, for a few protocols, the speedup is more modest: the verification time drops from about 4 to 1.1 minutes for the IKEv2-sigtomac protocol and from about 8.7 seconds to 1.7 seconds for IKEv2-mac protocol.

Falsification For rows marked by $\times$, line 2 corresponds to falsification time for the most abstract model. Falsification on the most abstract models is much faster than on the originals. For example, it takes 0.05 seconds to falsify the abstract model of IKEv1-pk-m for 8 threads, while it times out for the original one. In the unbounded case, the speedup factors are 7 for IKEv1-sig-m and 2180 for IKEv1-sig-m-perlman. We have manually analyzed the abstract attacks. Interestingly, none of them is spurious, suggesting that our measures to prevent them are effective. We expect that fast automatic detection of spurious attacks is feasible and will thus affect the current performance only negligibly.

Combination For the IKEv1-pk-m2 protocol, the tool verifies non-injective agreement for one role and falsifies it for the other one. Surprisingly, we obtain a remarkable speedup even though the analysis of this protocol is done three times (for two abstract and the original models). Our abstractions push the feasibility bound from 6 to 8 threads. As the property is verified very quickly for one role on the most abstract model, it needs to be analyzed only for the other role at lower abstraction levels. This explains the remarkable speedups we obtain and therefore illustrates an advantage of our abstraction mechanism in this case.

5 Related work and conclusions

Hui and Lowe [18] define several kinds of abstractions similar to ours with the aim of improving the performance of the CASPER/FDR verifier. They establish soundness only for ground messages and encryption with atomic keys. We work in

a more general model, cover additional properties, and treat the non-trivial issue of abstracting the open terms in protocol specifications. Other works [26,13,12] also propose a set of syntactic transformations, however without formally establishing their soundness. Using our results, we can, for instance, justify the soundness of the refinements in [13, Section 3.3]. Guttman [16,15] studies the preservation of security properties for a rich class of protocol transformations in the strand space model. His approach to property preservation is based on the simulation of protocol analysis steps instead of execution steps. Each such step explains the origin of a message. He does not have a syntactic soundness check.

In this work, we propose a set of syntactic protocol transformations that allows us to abstract realistic protocols and capture a large class of attacks. Unlike previous work in this area [23,18], our theory and soundness results accommodate equational theories, untyped variables, user-defined types, and subtyping. These features allow us to accurately model protocols, capture type-flaw attacks, and adapt to different verification tools, e.g., those supporting equational theories such as ProVerif and CL-atse. We have implemented our abstraction methodology for Scyther and validated its usefulness for on various IKE and ISO/IEC 9798 protocols. We also tested our technique (with manually produced abstractions) on ProVerif, CL-atse, OFMC, and SATMC. Our experiments clearly show that modern protocol verifiers can substantially benefit from our abstractions, which in many cases either enable the completion of verification tasks that have previously hit a resource limit or lead to dramatic speedups.

As mentioned earlier, our abstraction tool does not check for spurious attacks. We plan to complete the automation of the abstraction-refinement process by implementing this functionality. We are also interested in generalizing the tool and supporting more protocol verifiers. Finally, we intend to cover advanced security properties such as perfect forward secrecy.

References

1. A. Armando et al. The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures. In C. Flanagan and B. König, editors, *TACAS*, volume 7214 of *Lecture Notes in Computer Science*, pages 267–282. Springer, 2012.
2. M. Arapinis and M. Duflot. Bounding messages for free in security protocols. In V. Arvind and S. Prasad, editors, *FSTTCS*, volume 4855 of *Lecture Notes in Computer Science*, pages 376–387. Springer, 2007.
3. J. Arkko and H. Haverinen. RFC 4187: Extensible authentication protocol method for 3rd generation authentication and key agreement (EAP-AKA), 2006.
4. A. Armando and L. Compagna. SAT-based model-checking for security protocols analysis. *International Journal of Information Security*, 7(1):3–32, 2008.
5. D. A. Basin, S. Mödersheim, and L. Viganò. OFMC: A symbolic model checker for security protocols. *Int. J. Inf. Sec.*, 4(3):181–208, 2005.
6. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *CSFW*, pages 82–96. IEEE Computer Society, 2001.
7. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In R. M. Graham, M. A. Harrison, and R. Sethi, editors, *POPL*, pages 238–252. ACM, 1977.

8. C. Cremers. IKEv1 and IKEv2 protocol suites, 2011. https://github.com/cascremers/scyther/tree/master/gui/Protocols/IKE.
9. C. Cremers. ISO/IEC 9798 authentication protocols, 2012. https://github.com/cascremers/scyther/tree/master/gui/Protocols/ISO-9798.
10. C. J. F. Cremers. The Scyther tool: Verification, falsification, and analysis of security protocols. In A. Gupta and S. Malik, editors, *CAV*, volume 5123 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008.
11. C. J. F. Cremers. Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2. In V. Atluri and C. Díaz, editors, *ESORICS*, volume 6879 of *Lecture Notes in Computer Science*, pages 315–334. Springer, 2011.
12. A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Abstraction and refinement in protocol derivation. In *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW)*, 2004.
13. A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 13:423–482, 2005.
14. S. Escobar, C. Meadows, and J. Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In *FOSAD*, volume 5705 of *Lecture Notes in Computer Science*, pages 1–50. Springer, 2007.
15. J. D. Guttman. Transformations between cryptographic protocols. In P. Degano and L. Viganò, editors, *ARSPA-WITS*, volume 5511 of *LNCS*, pages 107–123. Springer, 2009.
16. J. D. Guttman. Security goals and protocol transformations. In *Theory of Security and Applications (TOSCA)*, volume 6993 of *LNCS*. Springer, 2011.
17. D. Harkins and D. Carrel. The Internet Key Exchange (IKE). IETF RFC 2409 (Proposed Standard), Nov 1998. Obsoleted by RFC 4306, updated by RFC 4109.
18. M. L. Hui and G. Lowe. Fault-preserving simplifying transformations for security protocols. *Journal of Computer Security*, 9(1/2):3–46, 2001.
19. J. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM J. Comput.*, 15(4):1155–1194, 1986.
20. C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet Key Exchange Protocol Version 2 (IKEv2). IETF RFC 5996, September 2010.
21. G. Lowe. A hierarchy of authentication specifications. In *IEEE Computer Security Foundations Workshop*, pages 31–43, Los Alamitos, CA, USA, 1997. IEEE Computer Society.
22. S. Meier, B. Schmidt, C. Cremers, and D. Basin. The TAMARIN prover for the symbolic analysis of security protocols. In N. Sharygina and H. Veith, editors, *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer, 2013.
23. B. T. Nguyen and C. Sprenger. Sound security protocol transformations. In D. A. Basin and J. C. Mitchell, editors, *POST*, volume 7796 of *Lecture Notes in Computer Science*, pages 83–104. Springer, 2013.
24. B. T. Nguyen and C. Sprenger. Abstractions for security protocol verification. Full version and experiments. Available from http://people.inf.ethz.ch/csprenge/post2015, October 2014.
25. L. Paulson. The inductive approach to verifying cryptographic protocols. *J. Computer Security*, 6:85–128, 1998.
26. D. Pavlovic and C. Meadows. Deriving secrecy in key establishment protocols. In *Proc. 11th European Symposium on Research in Computer Security (ESORICS)*, pages 384–403, 2006.
27. M. Turuani. The CL-Atse protocol analyser. In F. Pfenning, editor, *RTA*, volume 4098 of *Lecture Notes in Computer Science*, pages 277–286. Springer, 2006.

A Basic lemmas about the auxiliary functions and the type system

A.1 Lemma about splitting

Lemma 1. *For all $t, u \in \mathcal{T}(V, \Sigma_0)$ and all substitutions θ, $split(t) \subseteq split(u)$ implies that $split(t\theta) \subseteq split(u\theta)$*

Proof. Suppose that $split(t) \subseteq split(u)$ and $v \in split(t\theta)$. To show that $v \in split(u\theta)$ we distinguish two cases:

1. There is some $t' \in split(t)$ that is not a variable and $v = t'\theta$. Then $t' \in split(u)$ and thus t' is not a pair. Since t' is neither a variable nor a pair, we have $v \in split(u\theta)$.
2. There is a variable $X \in split(t)$ such that $v \in split(X\theta)$. Then $X \in split(u)$ and thus $v \in split(u\theta)$.

This completes the proof of the lemma. □

A.2 Lemmas about the type system

The subtyping relation respects the types' structures.

Lemma 2. *Let $\tau, \tau' \in \mathcal{Y}$ be such that $\tau \preccurlyeq \tau'$ and $\tau' \neq msg$. Then either*

(i) τ and τ' are atomic and $\tau \neq msg$, or
(ii) τ and τ' are composed and there are $n \geq 1$ and $g \in \Sigma^n$ such that $\tau = g(\tau_1, \ldots, \tau_n)$, $\tau' = g(\tau'_1, \ldots, \tau'_n)$, and $\tau_i \preccurlyeq \tau'_i$ for $i \in \widetilde{n}$.

Proof. We prove this lemma by rule induction on the derivation of $\tau \preccurlyeq \tau'$, depending on the last rule R that has been applied.

- $R = \mathrm{S}(msg)$: we have $\tau \in \mathcal{Y}$ and $\tau' = msg$, contradicting our assumption.
- $R = \mathrm{S}(\preccurlyeq_0)$: we have $\tau \preccurlyeq_0 \tau'$. Then it is clear that both τ and τ' are atomic and $\tau \neq msg$ by the definition of $\preccurlyeq_0$.
- $R = \mathrm{S}(refl)$: we have $\tau = \tau'$ and thus the conclusion holds trivially.
- $R = \mathrm{S}(trans)$: here, there is a τ'' such that $\tau \preccurlyeq \tau''$ and $\tau'' \preccurlyeq \tau'$. Since $\tau' \neq msg$, we derive (i) or (ii) from the induction hypothesis for $\tau'' \preccurlyeq \tau'$ to for τ'' and τ. In both cases, we have $\tau'' \neq msg$. Therefore, we can also apply the induction hypothesis to $\tau \preccurlyeq \tau''$. Hence, we either have that τ, τ'', and τ' are all atomic and $\tau \neq msg$ or they all have the same top-level constructor g and the arguments of τ and τ'' and of τ'' and τ' are in the subtyping relation and we conclude by applying $\mathrm{S}(trans)$ on the argument types.
- $R = \mathrm{S}(\Sigma^n)$: In this case, the conclusion (ii) follows directly from the rules' premises and conclusions.

□

The following lemma states that well-typed substitutions respect types.

Lemma 3. *Let θ be an R,Ax-normal substitution that is well-typed. Then for all terms $t \in \mathcal{T}$, we have $\Gamma(t\theta) \preccurlyeq \Gamma(t)$.*

Proof. The proof is proceeded by induction on t.

- If t is an atom then $t\theta = t$ and thus the lemma holds trivially.
- If t is a variable X then we distinguish two cases. If $X \notin dom(\theta)$ then we have $X\theta = X$ and this case holds trivially. Otherwise, we have $\Gamma(X\theta) \preccurlyeq \Gamma(X)$, since θ is well-typed and R,Ax-normal.
- If $t = c(t_1, \ldots, t_n)$ for some $c \in \Sigma^n$ and $n \geq 1$ then we have $t\theta = c(t_1\theta, \ldots, t_n\theta)$. Moreover, by induction hypothesis, we have $\Gamma(t_i\theta) \preccurlyeq \Gamma(t_i)$ for all $i \in \widetilde{n}$. This yields $\Gamma(t\theta) \preccurlyeq \Gamma(t)$ as required.

□

B Basic properties of typed abstractions

In this section, we prove several properties of typed abstractions. First, we show that two terms whose types are in a subtyping relation must be transformed by the same clause. Second, we describe the shapes of transformed terms in different cases. At the end, we prove that type inference is preserved under abstractions.

B.1 Uniform matching

The following lemma states that a term t matches a linear pattern p whenever t's type is a subtype of p's type.

Lemma 4. *Let $p \in \mathcal{P}$ be a linear pattern. Then, for all $t \in \mathcal{T}$ such that $\Gamma(t) \preccurlyeq \Gamma(p)$ there exists a substitution $\sigma : vars(p) \to \mathcal{T}$ such that $p\sigma = t$.*

Proof. We prove the lemma by induction on the structure of p. Below we use the abbreviations $\tau = \Gamma(t)$ and $\pi = \Gamma(p)$.

- If p is a pattern variable, then we define $\sigma = \{p \mapsto t\}$, hence $p\sigma = t$.
- If $p = g(p_1, \ldots, p_n)$ for $g \in \Sigma^n$, $n \geq 1$ then since $\Gamma(p) = \pi$, there exists $\pi_1, \ldots, \pi_n$ such that

  $$\pi = g(\pi_1, \ldots, \pi_n) \text{ and } \Gamma(p_i) = \pi_i \text{ for } i \in \widetilde{n}.$$

  Since $\tau \preccurlyeq \pi$, by Lemma 2, we have

  $$\tau = g(\tau_1, \ldots, \tau_n) \text{ and } \tau_i \preccurlyeq \pi_i \text{ for } i \in \widetilde{n}.$$

  Since $\Gamma(t) = \tau$ and τ is composed, t is not a variable. Therefore, we have

  $$t = g(t_1, \ldots, t_n) \text{ and } \Gamma(t_i) = \tau_i \text{ for } i \in \widetilde{n}.$$

  Hence, by induction hypothesis, there are $\sigma_i : vars(p_i) \to \mathcal{T}$ such that $t_i = p_i\sigma_i$ for $i \in \widetilde{n}$. Since p is linear, we can thus define $\sigma : vars(p) \to \mathcal{T}$ by $\sigma = \bigcup_{i=1}^{n} \sigma_i$. Hence, we obtain $p\sigma = t$.

This completes the proof of the lemma. □

Lemma 5 (Uniform matching). *Let $E_f = [f(p_1) = u_1, \ldots, f(p_n) = u_n]$ and*

$$matches(t) = \{i \in \widetilde{n} \mid \exists \theta.\, t = p_i\theta \ \wedge\ \Gamma(t) \preccurlyeq \Gamma(p_i)\}.$$

Then, for all $t, t' \in \mathcal{T}$ with $\Gamma(t') \preccurlyeq \Gamma(t)$, we have

(i) $matches(t) \subseteq matches(t')$,
(ii) $matches(t) = matches(t') = \{i\}$ *for some* $i \in \widetilde{n}$ *if* $\Gamma(t) \in \Pi_f{\downarrow}$.

In particular, $matches(t) = matches(\Gamma(t))$ for all terms $t \in \mathcal{T}$.

Proof. Let $t, t' \in \mathcal{T}$, $t : \tau$, $t' : \tau'$, and $\tau' \preccurlyeq \tau$. To see (i), suppose $i \in \mathit{matches}(t)$, i.e., $t = p_i\theta$ and $\Gamma(t) \preccurlyeq \Gamma(p_i)$ for some substitution θ. Since $\Gamma(t') \preccurlyeq \Gamma(t)$, we also have $\Gamma(t') \preccurlyeq \Gamma(p_i)$ and hence $i \in \mathit{matches}(t')$. This shows (i).

To see (ii), we first derive $\tau' \in \Pi_f{\downarrow}$ from the assumptions $\tau \in \Pi_f{\downarrow}$ and $\tau' \preccurlyeq \tau$. Therefore, there are $i, j \in \widetilde{n}$ and $\{\pi_i, \pi_j\} \subseteq \Pi_f$ such that $\tau \preccurlyeq \pi_i$ and $\tau' \preccurlyeq \pi_j$. Moreover, i and j are unique since F_f is pattern-disjoint. By Lemma 4 there are substitutions θ and θ' (with domains $\mathit{vars}(p_i)$ and $\mathit{vars}(p_j)$) such that $t = p_i\theta$ and $t' = p_j\theta'$. Hence, $\mathit{matches}(t) = \{i\}$ and $\mathit{matches}(t') = \{j\}$. Using the result in (i) derive $i = j$ as required. □

B.2 Shape lemma and termination

Lemma 6 (Shape lemma). *If $t \in \mathcal{T}$ then the following holds*

(i) If t is a variable or an atom, then $f(t) = t$.
(ii) If $t = c(t_1, \ldots, t_n)$, $c \in \Sigma^n \setminus (\mathsf{C_{Ax}} \cup \mathsf{C_{Key}})$ then we have

$$f(c(t_1, \ldots, t_n)) = \langle u_1, \ldots, u_d \rangle$$

for some $d > 0$ and for all $i \in \widetilde{d}$, u_i is one of the following forms:
(a) $u_i = c(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_n}))$ such that

$$\mathit{split}(\overline{v_j}) \subseteq \mathit{split}(t_j) \text{ for all } j \in \widetilde{n},\ c \neq \langle \cdot, \cdot \rangle, \text{ and}$$
$$c \in \mathsf{C_R} \Rightarrow \overline{v_n} = [t_n]$$

(b) $u_i = f(v)$ such that $\mathit{split}(v) \subseteq \mathit{split}(t_j)$ for some $j \in \widetilde{n}$. and it holds that

$$\forall j \in \mathit{pp}(c).\ \mathit{split}(t_j) \subseteq \mathit{split}(P(j))$$
$$\forall j \in \widetilde{n}.\ P(j) \subseteq \mathit{subs}(t) \setminus \{t\}$$

where

$$P(j) = \bigcup\{\mathit{set}(\overline{v_j}) \mid c(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_n})) \in \{u_1, \ldots, u_d\}\} \cup \bigcup\{v \mid f(v) \in \{u_1, \ldots, u_d\}\}$$

(iii) If $t = c(t_1, \ldots, t_n)$ for $c \in \Sigma^n \cap (\mathsf{C_{Ax}} \cup \mathsf{C_{Key}})$ and $n \geq 1$ then

$$f(t) = c(f(t_1), \ldots, f(t_n))$$

Proof. We prove this lemma by case distinction on the shape of the term $t \in \mathcal{T}$. We know that there exists the first pattern $f(p) = u$ in the list E_f^1 such that $\Gamma(t) \preccurlyeq \Gamma(p)$. By Lemma 4, there is a substitution θ such that $p\theta = t$. Hence, by Program 1, we have

$$f(t) = u\theta. \tag{2}$$

Case (i) where t is a variable or an atom follows immediately from Program 1 and the definition of E_f^0. Suppose t is composed. We distinguish the following cases.

– $t = c(t_1, \ldots, t_n)$ and $c \in \Sigma^n \setminus (\mathsf{C_{Ax}} \cup \mathsf{C_{Key}})$: since p is not a pattern variable and $t = p\theta$, we must have $p = c(p_1, \ldots, p_n)$. By Definition 6, we have $f(p) = \langle e_1, \ldots, e_d \rangle$ for some $d > 0$ and for all $i \in \widetilde{d}$, e_i is one of the following forms:
  1. $e_i = c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n}))$ such that

$$\begin{array}{l} \mathit{set}(\overline{q_j}) \subseteq \mathit{split}(p_j) \text{ for all } j \in \widetilde{n},\ c \neq \langle \cdot, \cdot \rangle, \text{ and} \\ c \in \mathsf{C_R} \Rightarrow \overline{q_n} = [p_n] \end{array}$$

  2. $e_i = f(q)$ such that $q \in \mathit{split}(p_j)$ for some $j \in \widetilde{n}$.

  and it holds that $\forall j \in \mathit{pp}(c).\ \mathit{split}(p_j) \subseteq Q(j)$ where

$$\begin{array}{rl} Q(j) = & \bigcup\{\mathit{set}(\overline{q_j}) \mid c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n})) \in \{e_1, \ldots, e_d\}\} \\ & \bigcup\{q \mid f(q) \in \{e_1, \ldots, e_d\}\} \end{array}$$

  Let $u_i = e_i\theta$ for all $i \in \widetilde{d}$. For each $i \in \widetilde{d}$, we distinguish two cases depending on the shape of e_i.
  - If $e_i = c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n}))$ for some vectors $\overline{q_1}, \ldots, \overline{q_n}$, then let $\overline{v_i} = \overline{q_i}\theta$. We obtain that $u_i = c(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_n}))$. From the fact that

$$\begin{array}{l} \mathit{set}(\overline{q_j}) \subseteq \mathit{split}(p_j) \text{ for all } j \in \widetilde{n}, \\ c \in \mathsf{C_R} \Rightarrow \overline{q_n} = [p_n], \text{ and } p_j\theta = t_j \text{ for all } j \in \widetilde{n}. \end{array}$$

    we derive that $c \in \mathsf{C_R} \Rightarrow \overline{v_n} = [t_n]$. Moreover, we also have

$$\mathit{split}(\overline{v_j}) \subseteq \mathit{split}(t_j) \text{ for all } j \in \widetilde{n}. \tag{3}$$

  - If $e_i = f(q)$ and $q \in \mathit{split}(p_j)$ for some $j \in \widetilde{n}$, then let $v = q\theta$. We derive that $\mathit{split}(v) \subseteq \mathit{split}(p_j\theta)$. Since $p_j\theta = t_j$, we obtain that

$$\mathit{split}(v) \subseteq \mathit{split}(t_j) \tag{4}$$

  It remains to show that

$$\begin{array}{l} \forall j \in \mathit{pp}(c).\ \mathit{split}(t_j) \subseteq \mathit{split}(P(j)) \\ \forall j \in \widetilde{n}.\ P(j) \subseteq \mathit{subs}(t) \setminus \{t\} \end{array}$$

  To see the first point, let $j \in \mathit{pp}(c)$. By Definition 6, we have $\mathit{split}(p_j) \subseteq Q(j)$. This implies $\mathit{split}(p_j\theta) \subseteq \mathit{split}(Q(j)\theta)$. Since $p_j\theta = t_j$ and $P_j = Q_j\theta$, we obtain $\mathit{split}(t_j) \subseteq P(j)$ as required. The second point follows immediately from (3) and (4).
– $t = c(t_1, \ldots, t_n)$ for $c \in \Sigma^n \cap (\mathsf{C_{Ax}} \cup \mathsf{C_{Key}})$ and $n \geq 1$: Since $p\theta = t$ and p is not a pattern-variable, we must have $p = c(q_1, \ldots, q_n)$ and $q_i\theta = t_i$ for all $i \in \widetilde{n}$. By Definition 6, we have

$$f(p_i) = c(f(q_1), \ldots, f(q_n))$$

  Hence we derive $f(t) = u_i\theta = c(f(t_1), \ldots, f(t_n))$ as required.

This completes the proof of the lemma. □

Proposition 1 (Termination). *The function f defined by Program 1 terminates on all terms $t \in \mathcal{T}$.*

Proof. We prove this by induction on the size of t. If $\Gamma(t)$ is an atom then the termination of $f(t)$ is immediate. If $\Gamma(t)$ is composed then from Lemma 6 we know that that f is called recursively on subterms of t. Hence, these calls terminate by the induction hypothesis. Therefore, $f(t)$ also terminates. This completes the proof of the proposition. □

Next, we prove that all abstracted protocols are protocols. This result enables chaining different abstractions to obtain more complex one.

Proposition 2. *$f(P)$ is also a protocol.*

Proof. Note that f maps roles to roles and is an identity on variables. Hence by Definition **??**, it is clear that $f(t)$ is a protocol. □

B.3 Lemma about abstracted types

Lemma 7. *Let σ be an R,Ax-normal ground substitution that is well-typed. Then $f(\sigma)$ is well-typed.*

Proof. Let $X \in dom(f(\sigma))$. Then we have $X \in dom(\sigma)$ and $f(X) = X$. Since σ is R,Ax-normal, so is $X\sigma$. Let t be a term such that $t =_{Ax} (Xf(\theta))\downarrow_{R,Ax}$. We need to show that $\Gamma(t) \preccurlyeq \Gamma(X)$. We consider two cases.

- If $\Gamma(X) = msg$ then it is trivial that $\Gamma(t) \preccurlyeq \Gamma(X)$.
- If $\Gamma(X) = \tau$ for an atomic type τ, then since σ is well-typed and $X\sigma$ is R,Ax-normal, it follows that $X\sigma$ is an atom. Thus, we have $f(X\sigma) = Xf(\sigma) = X\sigma$. Hence $t = X\sigma$. This implies $\Gamma(t) \preccurlyeq \Gamma(X)$ as required.

This completes the proof of the lemma. □

B.4 Lemma about splitting and intruder deducibility

Lemma 8. *Let $t, u \in \mathcal{T}$ such that $split(u) \subseteq split(t)$. Then we have*

$$split(f(u)) \subseteq split(f(t)).$$

Proof. We proceed by induction on $|u| + |t|$.

- If $|split(u)| + |split(t)| = 2$ then $split(u) \subseteq split(t)$ implies that $u = t$. Thus the lemma holds for this case.
- Now we assume that $|split(u)| + |split(t)| > 2$. There are two cases.
  - If u is not a pair then $split(u) = \{u\}$. Hence we have

$$u \in split(t) \tag{5}$$

Since $|split(u)| + |split(t)| > 2$, we have $t = \langle u_1, u_2 \rangle$. Hence by Lemma 6, we have $f(t) = \widehat{f}(\overline{v})$ for some vector $\overline{v}$ such that $split(t) = split(\overline{v})$.

By (5), there is $t' \in set(\overline{v})$ such that $u \in split(t')$. Moreover, we have $|t'| < |t|$. Thus by induction hypothesis, we have

$$split(f(u)) \subseteq split(f(t')).$$

Since $split(f(t')) \subseteq split(f(t))$, this implies that

$$split(f(u)) \subseteq split(f(t)).$$

- If $u = \langle u_1, u_2 \rangle$ then by Lemma 6, we have that $f(u) = \widehat{f}(\overline{r})$ for some vector $\overline{r}$ of length m such that $split(u) = split(\overline{r})$. Since $split(u) \subseteq split(t)$, we have $split(r_i) \subseteq split(t)$ for all $i \in \widetilde{m}$. Moreover, we also have that $|r_i| < |u|$. Hence by induction hypothesis, we have $split(f(r_i)) \subseteq split(f(t))$. Therefore, we obtain $split(f(u)) \subseteq split(f(t))$ as required.

This completes the proof of the lemma. □

The following lemma is an immediate corollary of Lemma 8.

Corollary 1. *Let $t \in \mathcal{T}$ and $u \in split(t)$. Then we have*

$$f(t) \vdash_E f(u)$$

The following lemma shows that if the intruder learns all the transformed components of a term, he can also learn the transformed term.

Lemma 9. *Let $T \cup \{u\} \subseteq \mathcal{T}$. Suppose $T \vdash_E f(t)$ for all $t \in split(u)$. Then $T \vdash_E f(u)$.*

Proof. We prove the lemma by induction on the size of u. If u is not a pair then $split(u) = \{u\}$ and $T \vdash_E f(u)$ follows immediately from the assumption. Otherwise, $u = \langle u_1, u_2 \rangle$. Then, by Lemma 6, we derive that

$$f(u) = \widehat{f}(\overline{r})$$

for some vector $\overline{r} = [r_1, \ldots, r_m]$ such that $split(\overline{r}) = split(u)$ and $set(\overline{r}) \subseteq subs(u) \setminus \{u\}$. Since u is R,Ax-normal, so are the r_i. Let $i \in \widetilde{m}$. By assumption and since $split(r_i) \subseteq split(u)$, we have $T \vdash_E f(t)$ for all $t \in split(r_i)$. Since $r_i \in subs(u) \setminus \{u\}$, we obtain $T \vdash_E f(r_i)$ from the induction hypothesis. Hence, the desired $T \vdash_E f(u)$ follows from $T \vdash_E f(r_i)$ for all $i \in \widetilde{m}$. □

The following lemma is a consequence of the two previous lemmas.

Lemma 10. *For all terms $t, u \in \mathcal{T}$, we have $split(t) \subseteq split(u)$ implies that $f(u) \vdash_E f(t)$.*

Proof. By Corollary 1, we have $f(u) \vdash_E f(p)$ for all $p \in split(u)$. Moreover, since $split(t) \subseteq split(u)$, we have $f(u) \vdash_E f(q)$ for all $q \in split(t)$. Hence, by Lemma 9, we have $f(u) \vdash_E f(t)$. □

C Substitution property

Theorem (Substitution property; Justification of Theorem 1). *Let θ be a substitution. Let $t \in \mathcal{T}$ be such that $t \in udom(F_f)$. If one of the following holds*

(i) f is homomorphic for $ct(t)$, or
(ii) $t \in udom(F_f)$ and θ is R,Ax-normal and well-typed,

then we have $f(t\theta) = f(t)f(\theta)$.

Proof. We prove the theorem by induction on the size of t. Suppose $E_f = [f(p_1) = u_1, \ldots, f(p_n) = u_n]$ and let t be a term such that $t \in udom(F_f)$. We distinguish two cases. If $\Gamma(t) = msg$ then t is a variable and thus $f(t) = t$ using the final identity fall-back clause of E_f^0. It follows that $f(t)f(\theta) = tf(\theta) = f(t\theta)$ as required. Otherwise, we have $\Gamma(t) \neq msg$. Let $t : \tau$ and $t\theta : \tau'$.

If condition (i) holds then t and $t\theta$ are abstracted by the same clause in E_f.

If condition (ii) holds then we have $\tau' \preccurlyeq \tau$ by Lemma 3. Since $t \in udom(F_f)$, we have $\tau \in \Pi_f{\downarrow}$. Hence, by Lemma 5, t and $t\theta$ are abstracted in the same way.

Thus, in both cases, there exists an unique pattern $(f(p) = u) \in E_f$ and substitutions θ' and θ'' such that $p\theta' = t$ and $p\theta'' = t\theta$. Thus, we also have $t\theta = p\theta'' = p\theta'\theta$. By Program 1 (modulo renamings), we have

$$f(t) = u[f/f_0]\theta' \quad \text{and} \quad f(t\theta) = u[f/f_0]\theta'\theta.$$

We distinguish two base cases.

- $u = p$ and $t = a$ is an atom. Then we obtain that $f(a\theta) = f(a) = a = af(\theta) = f(a)f(\theta)$ as required.
- $u = p$ and $t = X$ is a variable. Then we have $f(X) = X$. Let us consider two cases:
  - If $X \in dom(\theta)$, then we have
  $$f(X\theta) = Xf(\theta) = f(X)f(\theta).$$
  - If $X \notin dom(\theta)$, then since $dom(f(\theta)) = dom(\theta)$, we have $X \notin dom(f(\theta))$. Hence, we have
  $$f(X\theta) = f(X) = X = Xf(\theta) = f(X)f(\theta).$$

For the inductive cases, note that recursive calls of f have subterms of t as arguments by Lemma 6. Moreover, since $t \in udom(F_f)$, we also have $t' \in udom(F_f)$ for each term t' occurring as the argument of a recursive call of f in the computation of $f(t)$. This enables the application of the induction hypotheses (IH) below. We distinguish the following cases.

- $p = c(p_1, \ldots, p_n)$ for $c \in \Sigma^n \setminus (\mathsf{C_{Ax}} \cup \mathsf{C_{Key}})$, $n \geq 1$. In this case, we have
  $$f(c(p_1, \ldots, p_n)) = \langle e_1, \ldots, e_d \rangle$$
  for some $d > 0$ and for all $i \in \widetilde{d}$, e_i is one of the following forms:

1. $e_i = c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n}))$ such that

$$\begin{array}{l} \mathit{set}(\overline{q_j}) \subseteq \mathit{split}(p_j) \text{ for all } j \in \widetilde{n},\ c \neq \langle \cdot, \cdot \rangle, \text{ and} \\ c \in \mathsf{C_R} \Rightarrow \overline{q_n} = [p_n] \end{array}$$

2. $e_i = f(q)$ such that $q \in \mathit{split}(p_i)$ for some $j \in \widetilde{n}$.

and it holds that $\forall j \in pp(c).\ \mathit{split}(p_j) \subseteq Q(j)$ where

$$\begin{array}{rl} Q(j) = & \bigcup\{\mathit{set}(\overline{q_j}) \mid c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n})) \in \{e_1, \ldots, e_d\}\} \\ & \bigcup\{q \mid f(q) \in \{e_1, \ldots, e_d\}\} \end{array}$$

Therefore, we have

$$\begin{array}{c} f(t) = \langle e_1\theta', \ldots, e_d\theta' \rangle \\ f(t\theta) = \langle e_1\theta'\theta, \ldots, e_d\theta'\theta) \rangle \end{array}$$

This implies $f(t)f(\theta) = \langle e_1\theta' f(\theta), \ldots, e_d\theta' f(\theta) \rangle$. To see that $f(t\theta) = f(t)f(\theta)$, it is sufficient to show that $e_i\theta'\theta = e_i\theta' f(\theta)$ for all $i \in \widetilde{d}$. Let $i \in \widetilde{d}$, we distinguish two cases.

- If $e_i = c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n}))$ then we have

$$e_i\theta'\theta = c(\widehat{f}(\overline{q_1}\theta'\theta), \ldots, \widehat{f}(\overline{q_n}\theta'\theta))$$

  Since $\mathit{set}(\overline{q_j}) \subseteq \mathit{split}(p_j)$ for all $j \in \widetilde{n}$, we derive that $\mathit{set}(\overline{q_j}\theta') \subseteq \mathit{subs}(p_j\theta')$ for all $j \in \widetilde{n}$. Moreover, we have $p_j\theta' \in \mathit{subs}(t) \setminus \{t\}$. Therefore, by induction hypothesis, we derive that $\widehat{f}(\overline{q_j}\theta'\theta) = \widehat{f}(\overline{q_j}\theta')f(\theta)$ for all $j \in \widetilde{n}$. Note that for all $j \in \widetilde{n}$, we have $\widehat{f}(\overline{q_j}\theta') = \widehat{f}(\overline{q_j})\theta'$. Hence, we conclude that $e_i\theta'\theta = e_i\theta' f(\theta)$ as desired.

– $p = c(p_1, \ldots, p_n)$ for $c \in \Sigma^n \cap (\mathsf{C_{Ax}} \cup \mathsf{C_{Key}})$ and $n \geq 1$. In this case, we have

$$\begin{array}{c} f(p) = c(f(p_1), \ldots, f(p_n)) \\ f(t) = c(f(p_1\theta'), \ldots, f(p_n\theta')) \end{array}$$

Let $v_j = p_j\theta'$ for $j \in \widetilde{n}$. We prove this case by the following calculation:

$$\begin{array}{rll} f(t\theta) & = g(f(v_1\theta), \ldots, f(v_n\theta)) & \\ & = g(f(v_1)f(\theta), \ldots, f(v_n)f(\theta)) & \text{by IH} \\ & = g(f(v_1), \ldots, f(v_n))f(\theta) & \\ & = f(t)f(\theta) & \end{array}$$

This completes the proof of the theorem. □

D Deducibility preservation

Notation. For the sake of a lighter notation, we will omit set braces in intruder derivations and write, e.g., $T, t \vdash_E u$ instead of $T \cup \{t\} \vdash_E u$ for a set of terms T and individual terms t and u. We also write $T \vdash_E U$ for a set of terms U to mean that all terms in U are derivable from those in T. In the remains of this section, we assume that F_f is R, Ax-closed.

D.1 Preservation results for equality and reduction

Proposition 3 (Ax-equality preservation). *Let $t, u \in \mathcal{T}$. Then $t =_{Ax} u$ implies $f(t) =_{Ax} f(u)$.*

Proof. We prove this by induction on size the derivation $t =_{Ax} u$ depending on the last rule that has been applied.

- Reflexivity: In this case, we have $t = u$. Thus the lemma holds trivially.
- Axiom: In this case, there are a pair $\{s_1, s_2\} \in Ax$ and a substitution σ such that $t = s_1\sigma$ and $u = s_2\sigma$. Since F_f is R, Ax-closed, we know that $s_1, s_2 \in udom(F_f)$. Moreover, by point (v) in Definition 6, we know that f is homomorphic for $ct(s_1) \cup ct(s_2)$. By Theorem 1, we derive

$$f(t) = f(s_1)f(\sigma) = s_1 f(\sigma)$$
$$f(u) = f(s_2)f(\sigma) = s_2 f(\sigma)$$

  Thus by rule Axiom, we obtain that $s_1 f(\sigma) =_{Ax} s_2 f(\sigma)$. Therefore, we have $f(t) =_{Ax} f(u)$.
- Congruence: Suppose that $t = c(t_1, \ldots, t_n)$ for some $c \in \Sigma^n$. Then we have $u = g(u_1, \ldots, u_n)$ for some terms u_i, $i \in \widetilde{n}$ and

$$t_i =_{Ax} u_i \text{ for all } i \in \widetilde{n}. \tag{6}$$

  Moreover, since F_f is R, Ax-closed, we know that t and u match the same clause $f(p) = q$ in E_f^+. Hence there are substitutions θ, θ' such that $t = p\theta$ and $u = p\theta'$. We consider different shapes of p.
  - $p = c(p_1, \ldots, p_n)$ for $c \in \Sigma^n \setminus (\mathsf{C_{Ax}} \cup \mathsf{C_{Key}})$ and $n \geq 1$. In this case, we have

$$f(p) = \langle e_1, \ldots, e_d \rangle$$

    for some $d > 0$ for all $i \in \widetilde{d}$, e_i is one of the following forms:
    1. $e_i = c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n}))$ such that

$$set(\overline{q_j}) \subseteq split(p_j) \text{ for all } j \in \widetilde{n},\ c \neq \langle \cdot, \cdot \rangle, \text{ and}$$
$$c \in \mathsf{C_R} \Rightarrow \overline{q_n} = [p_n]$$

    2. $e_i = f(q)$ such that $q \in split(p_i)$ for some $j \in \widetilde{n}$.

and it holds that $\forall j \in pp(c).\ split(p_j) \subseteq Q(j)$ where

$$Q(j) = \bigcup\{set(\overline{q_j}) \mid c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n})) \in \{e_1, \ldots, e_d\}\} \\ \bigcup\{q \mid f(q) \in \{e_1, \ldots, e_d\}\}$$

Hence we have

$$f(t) = \langle e_1\theta, \ldots, e_d\theta\rangle \\ f(u) = \langle e_1\theta', \ldots, e_d\theta'\rangle$$

To see that $f(t) =_{Ax} f(u)$, it is sufficient to show that $e_i\theta =_{Ax} e_i\theta'$ for all $i \in \widetilde{d}$. Let $i \in \widetilde{d}$. We distinguish two cases depending on the shape of e_i.

- If $e_i = c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n}))$ then by (6), we derive that there is a sub-derivation $e\theta =_{Ax} e\theta'$, for all $j \in \widetilde{d}$ and all $e \in set(q_j)$. Hence, by induction hypothesis, we know that $f(e\theta) =_{Ax} f(e\theta')$ for all $j \in \widetilde{d}$ and all $e \in set(q_j)$. This implies $e_i\theta =_{Ax} e_i\theta'$ as desired.
- If $e_i = f(q)$ where $q \in split(p_j)$ for some $j \in \widetilde{n}$, then by a similar reasoning as before, we derive that there is a sub-derivation $q\theta =_{Ax} q\theta'$. By induction hypothesis, we have $f(q\theta) =_{Ax} f(q\theta')$ which implies $e_i\theta =_{Ax} e_i\theta'$ as desired.

- $p = c(p_1, \ldots, p_n)$ for $c \in \Sigma^n \cap (\mathsf{C_{Ax}} \cup \mathsf{C_{Key}})$, $n \geq 1$. Thus, we have

$$f(p) = c(f(p_1), \ldots, f(p_n)) \\ f(t) = c(f(t_1), \ldots, f(t_n)) \\ f(u) = c(f(u_1), \ldots, f(u_n))$$

By the induction hypothesis, we have $f(t_i) =_{Ax} f(u_i)$ for all $i \in \widetilde{n}$. It follows that $f(t) =_{Ax} f(u)$ as required.

– Transitivity: In this case, there is a term t' such that $t =_{Ax} t'$ and $t' =_{Ax} u$. By induction hypothesis, we have $f(t) =_{Ax} f(t')$ and $f(t') =_{Ax} f(u)$. Therefore, we have $f(t) =_{Ax} f(u)$.

This completes the proof of the lemma. □

Lemma 11. *Let $t \in rdom(F_f)$ be a term. Suppose that there is a term u such that $t \to_{R,Ax} u$. Then we have $f(t) =_E f(u)$.*

Proof. From the assumption, we know that there are a rewriting rule $l \to r \in R$, a position p, and a substitution σ such that $t|_p =_{Ax} l\sigma$ and $t \to_{R,Ax} t[r\sigma]_p$ and $u = t[r\sigma]_p$. By Proposition 3, we have $f(t|_p) =_{Ax} f(l\sigma)$. Moreover, since F_f is R, Ax-closed, we know that $l \in udom(F_f)$. Therefore, by Theorem 1, we have $f(l\sigma) = f(l)f(\sigma)$. By point (i) in Definition 9, we know that F_f is homomorphic for $ct(l)$. Therefore, we derive that $f(l) = l$ and obtain

$$f(t|_p) =_{Ax} lf(\sigma) \tag{7}$$

Since $t \in rdom(F_f)$, by (ii) in Definition 9, we have $f(t)|_p = f(t|_p)$. Hence, we have $f(t) =_{Ax} f(t)[lf(\sigma)]_p$ and therefore

$$f(t) \to_{R,Ax} f(t)[rf(\sigma)]_p \tag{8}$$

Since r is either a variable or an atom, we have $rf(\sigma) = f(r\sigma)$. Note that $f(t)[f(r\sigma)]_p = f(t[r\sigma]_p)$. Therefore, we derive $f(t)[rf(\sigma)]_p = f(t[r\sigma]_p)$. Together with (8), we obtain

$$f(t) \to_{R,Ax} f(t[r\sigma]_p).$$

Hence, $f(t) =_E f(u)$ as required. □

Proposition 4 (Reduction preservation). *Let $t \in rdom(F_f)$ be a term. Then we have $f(t) =_E f(t{\downarrow_{R,Ax}})$.*

Proof. We proceed by induction on the length of the rewriting sequence

$$t = t_0 \to_{R,Ax} t_1 \to_{R,Ax} \cdots \to_{R,Ax} t_m =_{Ax} t{\downarrow_{R,Ax}} .$$

Without loss of generality, we assume that the rewriting is done inside-out. The base case ($m = 0$) can be derived directly from using Proposition 3.

In the inductive case ($m > 0$), there are a rewriting rule $l \to r \in R$, a position p, and a substitution σ such that $t|_p =_{Ax} l\sigma$ and $t \to_{R,Ax} t[r\sigma]_p = t_1$. Thus, by Lemma 11, we derive $f(t) =_E f(t_1)$. Note that $r\sigma \in subs(l\sigma)$. Together with the fact that the rewriting is done inside-out, this implies $r\sigma$ is R,Ax-normal. Hence, we have $t[r\sigma]_p \in rdom(F_f)$. By the induction hypothesis and by Proposition 3, we have $f(t_1) =_E f(t_m)$ and $f(t_m) =_{Ax} f(t{\downarrow_{R,Ax}})$. Therefore, we obtain that $f(t) =_E f(t')$ as required. □

Theorem (Equality preservation; Justification of Theorem 2). *Let t and u be terms such that $\{t, u\} \subseteq rdom(F_f)$. Then $t =_E u$ implies $f(t) =_E f(u)$.*

Proof. Suppose $t =_E u$. Then $t{\downarrow_{R,Ax}} =_{Ax} u{\downarrow_{R,Ax}}$. By Proposition 3, we have $f(t{\downarrow_{R,Ax}}) =_{Ax} f(u{\downarrow_{R,Ax}})$. From Proposition 4 we have $f(t) =_E f(t{\downarrow_{R,Ax}})$ and $f(u) =_E f(u{\downarrow_{R,Ax}})$. Hence, $f(t) =_E f(u)$ as required. □

D.2 Preservation results for deducibility

Theorem (Deducibility preservation; Justification of Theorem 3). *Let $T \cup \{t\} \subseteq \mathcal{N}$ be a set of ground network messages such that $\mathcal{C} \subseteq T$ and T is R,Ax-normal. Then $T \vdash_E t$ implies $f(T) \vdash_E f(t{\downarrow_{R,Ax}})$.*

Proof. By induction on the derivation D of $T \vdash_E t$, depending on the last rule that has been applied in this derivation.

- Ax: In this case, we have $t \in T$. Therefore $f(t) \in f(T)$ and thus $f(T) \vdash_E f(t)$. Moreover, since T is R,Ax-normal, so is t. Thus by Proposition 3, we have $f(t) =_{Ax} f(t{\downarrow_{R,Ax}})$. Hence, we obtain $f(T) \vdash_E f(t{\downarrow_{R,Ax}})$ as required.
- Eq: We have $T \vdash_E t'$ and $t' =_E t$. By induction hypothesis, we have $f(T) \vdash_E f(t'{\downarrow_{R,Ax}})$. Moreover, since $t' =_E t$, we derive $t'{\downarrow_{R,Ax}} =_{Ax} t{\downarrow_{R,Ax}}$. By Proposition 3, we have $f(t'{\downarrow_{R,Ax}}) =_{Ax} f(t{\downarrow_{R,Ax}})$. It follows that $f(T) \vdash_E f(t{\downarrow_{R,Ax}})$ as required.
- Comp: In this case, $t = g(u_1, \ldots, u_m)$ for $g \in \Sigma^m$ and the rule's premises are $T \vdash_E u_i$ for $i \in \widetilde{m}$. The induction hypotheses are $f(T) \vdash_E f(u_i{\downarrow_{R,Ax}})$ for $i \in \widetilde{m}$. Note that $t{\downarrow_{R,Ax}} = (g(u_1{\downarrow_{R,Ax}}, \ldots, u_m{\downarrow_{R,Ax}})){\downarrow_{R,Ax}}$. Hence without loss of generality, we can assume that u_i are R,Ax-normal for all $i \in \widetilde{m}$. We distinguish two cases depending on whether t is R,Ax-normal or not.

Case 1: t is not R,Ax-normal. Since u_i are R,Ax-normal for all $i \in \widetilde{m}$, some rewrite rule $l \to r \in R$ can be applied at the root position of t, i.e., there is a substitution σ such that $t =_{Ax} l\sigma$. This rule is of one of the following types.

- R4: In this case, we have $t{\downarrow_{R,Ax}} = a$ for some constant a. Hence $f(t) = f(a) = a$. Since $\mathcal{C} \subseteq T$ and $\mathcal{C} = f(\mathcal{C})$, we have $f(T) \vdash_E f(t{\downarrow_{R,Ax}})$.
- R3: We have $t{\downarrow_{R,Ax}} =_{Ax} u_j$ for some $j \in \widetilde{m}$. By Proposition 3, we have $f(t{\downarrow_{R,Ax}}) =_{Ax} f(u_j)$ which implies $f(T) \vdash_E f(t{\downarrow_{R,Ax}})$ by the induction hypothesis.
- R1: We have $m = 2$ and

$$\begin{aligned} l &= g(c(x_1, \ldots, x_{n-1}, t'), u') \to x_j, \\ r &= x_j \text{ for some } j, 1 \leq j \leq n-1, \end{aligned}$$

where $x_1, \ldots, x_{n-1}$ are variables and t', u' are terms.
Let $t_i = x_i\sigma$ for all $i, 1 \leq i \leq n-1$ and $t_n = t'\sigma$. Then we have

$$\begin{aligned} u_1 &= c(t_1, \ldots, t_{n-1}, t_n), \\ u_2 &= u'\sigma. \end{aligned}$$

Moreover, since u_1 is R,Ax-normal, so is t_j. Therefore, we have $t{\downarrow_{R,Ax}} =_{Ax} t_j$. By Proposition 3, we know that $f(t{\downarrow_{R,Ax}}) =_{Ax} f(t_j)$. Therefore, to see that $f(T) \vdash_E f(t{\downarrow_{R,Ax}})$, it is sufficient to show that $f(T) \vdash_E f(t_j)$. From the induction hypothesis, we know that $f(T) \vdash_E f(u_1)$. By Lemma 6, we have

$$f(u_1) = \langle w_1, \ldots, w_d \rangle$$

for some $d > 0$ for all $i \in \widetilde{d}$, w_i is one of the following forms:

1. $w_i = c(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_n}))$ such that

$$\begin{aligned} &\mathit{split}(\overline{v_k}) \subseteq \mathit{split}(t_k) \text{ for all } k \in \widetilde{n},\ c \neq \langle \cdot, \cdot \rangle, \text{ and} \\ &c \in \mathsf{C_R} \Rightarrow \overline{v_n} = [t_n], \end{aligned}$$

2. $w_i = f(v)$ where $\mathit{split}(v) \subseteq \mathit{split}(t_k)$ for some $k \in \widetilde{n}$

and it holds that

$$\begin{aligned} &\forall k \in \mathit{pp}(c).\ \mathit{split}(t_k) \subseteq \mathit{split}(P(k)) \\ &\forall k \in \widetilde{n}.\ P(k) \subseteq \mathit{subs}(u_1) \setminus \{u_1\} \end{aligned} \tag{9}$$

where

$$\begin{aligned} P(k) = \bigcup\{&\mathit{set}(\overline{v_k}) \mid c(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_n})) \in \{w_1, \ldots, w_d\}\} \\ \bigcup\{&v \mid f(v) \in \{w_1, \ldots, w_d\}\} \end{aligned}$$

By (9) and Lemma 10, it is sufficient to establish $f(T) \vdash_E f(e)$ for all $e \in P(j)$ in order to conclude the desired $f(T) \vdash_E f(t_j)$.
Let $e \in P(j)$. Note that there is $k \in \widetilde{d}$ such that one of the following holds.

$$\begin{aligned} &w_k = c(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_n})) \text{ and } e \in \mathit{set}(\overline{v_j}) \\ &w_k = f(v) \text{ and } e = v \end{aligned}$$

Clearly, we can derive $f(T) \vdash_E w_k$ from $f(T) \vdash_E f(u_1)$ by projections. We have

$$f(T) \vdash_E c(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_n})) \tag{10}$$

In the second case, we immediately obtain $f(T) \vdash_E f(e)$. In the first case, we know that $\overline{v_n} = [t_n]$ by Lemma 6. Hence, we have

$$\widehat{f}(\overline{v_n}) = f(t_n) = f(t'\sigma) \tag{11}$$

We also have $f(u_2) = f(u'\sigma)$. By Theorem 1, we derive

$$\begin{array}{l} f(t_n) = f(t')f(\sigma) \\ f(u_2) = f(u')f(\sigma) \end{array}$$

Note that $ct(t') \cup ct(u') \subseteq \mathsf{C_{Key}}$. Hence, by Definition 6, we have $f(t') = t'$ and $f(u') = u'$. This yields

$$\begin{array}{l} f(t_n) = t'f(\sigma) \\ f(u_2) = u'f(\sigma) \end{array} \tag{12}$$

Moreover, by induction hypothesis, we have $f(T) \vdash_E f(u_2)$. Together with (10), (11), and (12), we derive

$$f(T) \vdash_E g(c(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_{n-1}}), t'f(\sigma)), u'f(\sigma)))$$

Since $(vars(t') \cup vars(u')) \cap \bigcup_{k=1}^{n-1} x_k = \emptyset$ and $x_p \neq x_q$ for all p, q such that $1 \leq p \neq q \leq n-1$, we can define a substitution θ as follows.

$$\begin{array}{l} x_q\theta = \widehat{f}(\overline{v_q}) \text{ for } 1 \leq q \leq n-1, \\ x\theta \ \ = xf(\sigma) \text{ for all } x \in vars(u') \cup vars(t'). \end{array}$$

Then we have $g(c(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_{n-1}}), t'f(\sigma)), u'f(\sigma))) = l\theta$. Thus, we derive $g(c(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_{n-1}}), t'f(\sigma)), u'f(\sigma))) \to_{R,Ax} \widehat{f}(\overline{v_j})$. This implies $f(T) \vdash_E \widehat{f}(\overline{v_j})$. Together with $e \in set(\overline{v_j})$ and Lemma 10, we derive that $f(T) \vdash_E f(e)$.

- R2: This case is treated similarly as the case for R1.

Case 2: t is R,Ax-normal. By Proposition 3, we have $f(t) =_{Ax} f(t{\downarrow_{R,Ax}})$. Hence, it is sufficient to show that $f(T) \vdash_E f(t)$. We do case analysis on g.

- If $g \in \Sigma^n \setminus (\mathsf{C_{Ax}} \cup \mathsf{C_{Key}})$ then by Lemma 6, we know that

$$f(t) = \langle w_1, \ldots, w_d \rangle$$

for some $d > 0$ and for all $i \in \widetilde{d}$, w_i is one of the following forms:

1. $w_i = g(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_m}))$ such that

$$\begin{array}{l} split(\overline{v_k}) \subseteq split(u_k) \text{ for all } k \in \widetilde{m},\ g \neq \langle \cdot, \cdot \rangle, \\ \text{and } g \in \mathsf{C_R} \Rightarrow \overline{v_m} = [u_m] \end{array}$$

2. $w_i = f(v)$ such that $split(v) \subseteq split(u_k)$ for some $k \in \widetilde{m}$ and it holds that

$$\forall k \in pp(g).\ split(u_k) \subseteq split(P(k))$$
$$\forall k \in \widetilde{m}.\ P(k) \subseteq subs(t) \setminus \{t\}$$

where

$$P(k) = \bigcup\{set(\overline{v_k}) \mid g(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_m})) \in \{w_1, \ldots, w_d\}\} \cup \bigcup\{v \mid f(v) \in \{w_1, \ldots, w_d\}\}$$

To see that $f(T) \vdash_E f(t)$, it is sufficient to show that $f(T) \vdash_E w_i$ for all $i \in \widetilde{d}$. Let $i \in \widetilde{d}$. We do a case distinction on the shape of w_i.

- If $w_i = c(\widehat{f}(\overline{v_1}), \ldots, \widehat{f}(\overline{v_m}))$ then since $split(\overline{v_j}) \subseteq split(u_j)$ for all $j \in \widetilde{m}$, by Lemma 10, we derive that $f(u_j \vdash_E \widehat{f}(\overline{v_j})$ for all $j \in \widetilde{m}$. Moreover, by induction hypothesis, we have $f(T) \vdash_E f(u_j)$ for all $j \in \widetilde{m}$. Therefore, we derive that $f(T) \vdash_E \widehat{f}(\overline{v_j})$ for all $j \in \widetilde{m}$. This implies $f(T) \vdash_E w_i$ as desired.
- If $w_i = f(v)$ for some term v such that $split(v) \subseteq split(u_k)$ for some $k \in \widetilde{m}$. By a similar reasoning as above, we conclude that $f(T) \vdash_E w_i$.

- For the remaining cases, we have $f(t) = g(f(u_1), \ldots, f(u_m))$. Then $f(T) \vdash_E f(t)$ follows immediately from the induction hypothesis.

This completes the proof of the theorem. □

We are now ready to establish deducibility preservation for open terms. This result is crucial to for reachability preservation.

Proposition 5 (Deducibility preservation for open terms). *Let σ be a well-typed R,Ax-normal ground substitution, T be a set of terms, and u be term such that*

(i) $f(IK_0) \subseteq IK_0'$,
(ii) $T \cup \{u\} \subseteq udom(F_f) \cap rdom(F_f)$.

Then $T\sigma, IK_0 \vdash_E u\sigma$ implies $f(T)f(\sigma), IK_0' \vdash_E f(u)f(\sigma)$.

Proof. We have

$T\sigma, IK_0$	$\vdash_E u\sigma$	by assumption
$\Rightarrow (T\sigma)\downarrow_{R,Ax}, IK_0$	$\vdash_E (u\sigma)\downarrow_{R,Ax}$	by rule Eq
$\Rightarrow f((T\sigma)\downarrow_{R,Ax}), f(IK_0)$	$\vdash_E f((u\sigma)\downarrow_{R,Ax})$	by Theorem 3
$\Rightarrow f((T\sigma)\downarrow_{R,Ax}), IK_0'$	$\vdash_E f((u\sigma)\downarrow_{R,Ax})$	by assumption (i)
$\Rightarrow f(T\sigma), IK_0'$	$\vdash_E f(u\sigma)$	by Theorem 2 and Eq
$\Rightarrow f(T)f(\sigma), IK_0'$	$\vdash_E f(u)f(\sigma)$	by Theorem 1

This completes the proof of the proposition. □

E Soundness of typed abstractions (Section 3.2)

E.1 Reachability preservation

Using the deducibility preservation for open terms, we show that each reachable state in the original protocol can be simulated by one in the abstracted protocol.

Theorem 5 (Reachability preservation). *Suppose that*

(i) $f(IK_0) \subseteq IK_0'$,
(ii) F_f *is* R, Ax*-closed,*
(iii) $\mathcal{M}_P \subseteq udom(F_f) \cap rdom(F_f)$.

Let (tr, th, σ) *be a reachable state of* P *such that* σ *is* R,Ax*-normal. Then* $(f(tr), f(th), f(\sigma))$ *is a reachable state of* $f(P)$.

Proof. Note that $f(\sigma)$ is a well-typed ground substitution by Lemma 7. We now show that $(f(tr), f(th), f(\sigma))$ is reachable in $f(P)$ by induction on the number n of transitions leading to a state (tr, th, σ).

– Base case ($n = 0$): For all $i \in dom(th)$, there exists $R \in dom(P)$ such that $th(i) = (R, P(R))$. Hence we have

$$f(th)(i) = (R, f(P(R))) = (R, f(P)(R)) \tag{13}$$

Since (ϵ, th, σ) is reachable, for all $v \in dom(P)$ and $i \in TID$ we have $v^{\#i}\sigma \in \mathcal{A}$. Moreover, we have $v^{\#i} f(\sigma) = v^{\#i}\sigma$, we also have

$$v^{\#i} f(\sigma) \in \mathcal{A} \tag{14}$$

By (13), (14) and $f(\epsilon) = \epsilon$, it is obvious that $(f(\epsilon), f(th), f(\sigma))$ is reachable in $f(P)$.

– Inductive case ($n = k + 1$): Suppose (tr', th', σ) is reachable in k steps and there is a transition $(tr', th', \sigma) \to (tr, th, \sigma)$. By induction hypothesis, we have

$$(f(tr'), f(th'), f(\sigma)) \text{ is reachable in } f(P) \tag{15}$$

We consider two cases according to the rule r that has been applied in step $k + 1$.

- If $r = SEND$ then there exists $i \in TID$ and $R \in dom(P)$ such that

$$\begin{aligned} th'(i) &= (R, \mathsf{snd}(pt).tl) \\ tr &= tr' \cdot (i, \mathsf{snd}(pt)) \\ th &= th'[i \mapsto (R, tl)] \end{aligned} \tag{16}$$

By (16) we have

$$\begin{aligned} f(tr) &= f(tr') \cdot (i, \mathsf{snd}(f(pt))) \\ f(th) &= f(th')[i \mapsto (R, f(tl))] \end{aligned} \tag{17}$$

By (16) we have

$$f(th')(i) = (R, \mathsf{snd}(f(pt)) \cdot f(tl)) \tag{18}$$

By (18), (16), (17) and rule *SEND*, we have

$$(f(tr'), f(th'), f(\sigma)) \to (f(tr), f(th), f(\sigma))$$

Together with (15) this implies that $(f(tr), f(th), f(\sigma))$ is reachable in $f(P)$.

- If $r = RECV$ then there exists $i \in TID$ and $R \in dom(P)$ such that

$$\begin{array}{l} th'(i) = (R, \mathsf{rcv}(u) \cdot tl) \\ IK(tr')\sigma, IK_0 \vdash_E u\sigma \end{array} \tag{19}$$

and

$$\begin{array}{l} tr = tr' \cdot (i, \mathsf{rcv}(u)) \\ th = th'[i \mapsto (R, tl)] \end{array} \tag{20}$$

By (19) and (20) we have

$$\begin{array}{l} f(tr) = f(tr') \cdot (i, \mathsf{rcv}(f(u))) \\ f(th) = f(th')[i \mapsto (R, f(tl))] \end{array}$$

To justify $(f(tr'), f(th'), f(\sigma)) \to (f(tr), f(th), f(\sigma))$, it is sufficient to establish the following two premises of rule *RECV*:

1. $f(th')(i) = (R, \mathsf{rcv}(f(u)) \cdot f(tl))$, which follows from (19), and
2. $IK(f(tr'))f(\sigma), IK_0' \vdash_E f(u)f(\sigma)$. This follows from (19), Proposition 5, and the fact that $f(IK(tr')) = IK(f(tr'))$.

Together with (15) this implies that $(f(tr), f(th), f(\sigma))$ is reachable in $f(P)$.

This completes the proof of the theorem. □

E.2 Soundness

We show in the following lemma that whenever a protocol admits an attack then there is an R,Ax-normal attack.

Lemma 12. *Let $\phi \in \mathcal{L}_P$ and let (tr, th, σ) be a reachable state of P. Then the following holds.*

(i) $(tr, th, \sigma{\downarrow_{R,Ax}})$ is a reachable state in P, and
(ii) if $(tr, th, \sigma) \not\models \phi$ then $(tr, th, \sigma{\downarrow_{R,Ax}}) \not\models \phi$.

Proof. Let $\sigma' = \sigma{\downarrow_{R,Ax}}$. Then σ' is well-typed. Next, we show reachability for σ', i.e., we need to show that (tr, th, σ') is reachable in P. We prove this by induction on the number n of transitions leading to (tr, th, σ).

– Base case ($n = 0$): Since (ϵ, th, σ) is reachable, so is (ϵ, th, σ').

– Inductive case ($n = k + 1$): Suppose (tr', th', σ) is reachable in k steps and there is a transition $(tr', th', \sigma) \to (tr, th, \sigma)$. By induction hypothesis, we have

$$(tr', th', \sigma) \text{ is reachable in } P. \tag{21}$$

We consider two cases according to the rule r that has been applied in step $k + 1$.

- If $r = \mathit{SEND}$ then it is obvious that $(tr', th', \sigma') \to (tr, th, \sigma')$. This by (28) yields that (tr, th, σ') is reachable in P.
- If $r = \mathit{RECV}$ then there exists $i \in \mathit{TID}$ and $R \in \mathit{dom}(P)$ such that $th'(i) = (R, \mathsf{rcv}(u) \cdot tl)$, $tr = tr' \cdot (i, \mathsf{rcv}(u))$, $th = th'[i \mapsto (R, tl)]$, and

$$\mathit{IK}(tr')\sigma, \mathit{IK}_0 \vdash_E u\sigma$$

Hence, we have $\mathit{IK}(tr')\sigma', \mathit{IK}_0 \vdash_E u\sigma'$. Thus the reachability of (tr, th, σ') in P follows immediately.

Finally, we show attack preservation for σ', i.e., we need to show that

$$\forall \vartheta.\ (tr, th, \sigma, \vartheta) \not\models \phi \Rightarrow (tr, th, \sigma', \vartheta) \not\models \phi$$

We prove this by induction on the structure of ϕ. Note that the base cases for the literals that do not depend on σ are trivial. Hence, it is enough to consider the following cases.

– $\phi \equiv m = m'$ or $\phi \equiv \neg(m = m')$.

$$\begin{aligned} & (tr, th, \sigma, \vartheta) \models m = m' \\ \Leftrightarrow\ & m\sigma =_E m'\sigma \\ \Leftrightarrow\ & m\sigma' =_E m'\sigma' \\ \Leftrightarrow\ & (tr, th, \sigma\downarrow_{R,Ax}, \vartheta) \models m = m' \end{aligned}$$

– $\phi \equiv \mathit{honest}(i, R)$ or $\phi \equiv \neg\mathit{honest}(i, R)$.

$$\begin{aligned} & (tr, th, \sigma, \vartheta) \models \mathit{honest}(i, R) \\ \Leftrightarrow\ & R^{\vartheta(i)}\sigma \in \mathcal{A}_H \\ \Leftrightarrow\ & R^{\vartheta(i)}\sigma' \in \mathcal{A}_H && \text{since } R^{\vartheta(i)}\sigma = R^{\vartheta(i)}\sigma' \\ \Leftrightarrow\ & (tr, th, \sigma', \vartheta) \models \mathit{honest}(i, R) \end{aligned}$$

– $\phi \equiv \mathit{secret}(m)$.

$$\begin{aligned} & (tr, th, \sigma, \vartheta) \not\models \mathit{secret}(m) \\ \Leftrightarrow\ & \mathit{IK}(tr)\sigma, \mathit{IK}_0 \vdash_E m\sigma \\ \Rightarrow\ & \mathit{IK}(tr)\sigma', \mathit{IK}_0 \vdash_E m\sigma' \\ \Leftrightarrow\ & (tr, th, \sigma', \vartheta) \not\models \mathit{secret}(m) \end{aligned}$$

The inductive cases are routines. □

E.3 Soundness of typed abstractions

Theorem (Soundness; Justification of Theorem 4). *Suppose P, ϕ, and F_f satisfy*

- $f(IK_0) \subseteq IK_0'$,
- F_f *is* R, Ax*-closed,*
- $\mathcal{M}_P \subseteq udom(F_f) \cap rdom(F_f)$, *and*
- ϕ *is safe for* P *and* f.

Then, for all states (tr, th, σ) reachable in P, we have

(i) $(f(tr), f(th), f(\sigma{\downarrow_{R,Ax}}))$ *is a reachable state of* $f(P)$,
(ii) $(tr, th, \sigma) \not\models \phi$ *implies* $(f(tr), f(th), f(\sigma{\downarrow_{R,Ax}})) \not\models f(\phi)$.

Proof. Let (tr, th, σ) be a reachable state of P. By Lemma 12, we know that (a) $(tr, th, \sigma\downarrow_{R,Ax})$ is a reachable state of P, and (b) $(tr, th, \sigma) \not\models \phi$ implies $(tr, th, \sigma\downarrow_{R,Ax}) \not\models \phi$. Let $\sigma' = f(\sigma\downarrow_{R,Ax})$. Then point (i) follows from (a) and Theorem 5. Using (b) we reduce point (ii) to showing that $(tr, th, \sigma\downarrow_{R,Ax}) \not\models \phi$ implies $(f(tr), f(th), \sigma') \not\models f(\phi)$, which we establish by proving the following generalized statement by induction on the structure of ϕ (which may now contain free thread-id variables).

$$\forall \vartheta.\ (tr, th, \sigma\downarrow_{R,Ax}, \vartheta) \not\models \phi \Rightarrow (f(tr), f(th), \sigma', \vartheta) \not\models f(\phi)$$

Note that a formula is safe if and only if all its subformulas are safe. The literals form the base cases of the induction. We cover all atoms and their negations (except $secret(m)$) in a single equivalence-based argument, where the right-to-left direction covers the positive literal and the other direction the corresponding negative literal. We remark that $(tr, th, \sigma\downarrow_{R,Ax}, \vartheta) \not\models A$ is equivalent to $(tr, th, \sigma\downarrow_{R,Ax}, \vartheta) \models \neg A$ for all atoms A (but not for all formulas, since $\mathcal{L}_P$ is not closed under negation).

- $\phi \equiv i = j$ or $\phi \equiv \neg(i = j)$.

$$\begin{aligned} & (tr, th, \sigma\downarrow_{R,Ax}, \vartheta) \models i = j \\ \Leftrightarrow\ & \vartheta(i) = \vartheta(j) \\ \Leftrightarrow\ & (f(tr), f(th), \sigma', \vartheta) \models f(i = j) \end{aligned}$$

- $\phi \equiv m = m'$.

$$\begin{aligned} & (tr, th, \sigma\downarrow_{R,Ax}, \vartheta) \models m = m' \\ \Rightarrow\ & m\sigma\downarrow_{R,Ax} =_E m'\sigma\downarrow_{R,Ax} \\ \Rightarrow\ & f(m\sigma\downarrow_{R,Ax}) =_E f(m'\sigma\downarrow_{R,Ax}) && \text{by Theorem 2} \\ \Rightarrow\ & f(m)f(\sigma\downarrow_{R,Ax}) = f(m')f(\sigma\downarrow_{R,Ax}) && \text{by Theorem 1} \\ \Leftrightarrow\ & f(m)\sigma' = f(m')\sigma' \\ \Leftrightarrow\ & (f(tr), f(th), \sigma', \vartheta) \models f(m) = f(m') \end{aligned}$$

– $\phi \equiv \neg(m = m')$.

$$
\begin{array}{ll}
& (f(tr), f(th), \sigma', \vartheta) \vDash f(m) = f(m') \\
\Rightarrow & f(m)\sigma' =_E f(m')\sigma' \\
\Leftrightarrow & f(m)f(\sigma{\downarrow_{R,Ax}}) =_E f(m')f(\sigma{\downarrow_{R,Ax}}) \\
\Rightarrow & f(m\sigma{\downarrow_{R,Ax}}) =_E f(m'\sigma{\downarrow_{R,Ax}}) & \text{by Theorem 1} \\
\Rightarrow & m\sigma{\downarrow_{R,Ax}} =_E m'\sigma{\downarrow_{R,Ax}} & \text{since } \phi \text{ is safe} \\
\Rightarrow & (tr, th, \sigma{\downarrow_{R,Ax}}, \vartheta) \vDash m = m'
\end{array}
$$

– $\phi \equiv \mathit{role}(i, R)$ or $\phi \equiv \neg\mathit{role}(i, R)$.

$$
\begin{array}{ll}
& (tr, th, \sigma{\downarrow_{R,Ax}}, \vartheta) \vDash \mathit{role}(i, R) \\
\Leftrightarrow & \exists \mathit{seq} \in \mathit{Evt}^*.\ \mathit{th}(\vartheta(i)) = (R, \mathit{seq}) \\
\Leftrightarrow & \exists \mathit{seq} \in \mathit{Evt}^*.\ f(th)(\vartheta(i)) = (R, f(\mathit{seq})) \\
\Leftrightarrow & (f(tr), f(th), \sigma', \vartheta) \vDash \mathit{role}(i, R)
\end{array}
$$

– $\phi \equiv \mathit{honest}(i, R)$ or $\phi \equiv \neg\mathit{honest}(i, R)$.

$$
\begin{array}{ll}
& (tr, th, \sigma{\downarrow_{R,Ax}}, \vartheta) \vDash \mathit{honest}(i, R) \\
\Leftrightarrow & R^{\vartheta(i)}\sigma{\downarrow_{R,Ax}} \in \mathcal{A}_H \\
\Leftrightarrow & R^{\vartheta(i)}f(\sigma{\downarrow_{R,Ax}}) \in \mathcal{A}_H & f \text{ is the identity on } \mathcal{A} \\
\Leftrightarrow & R^{\vartheta(i)}\sigma' \in \mathcal{A}_H & \text{since } R^{\vartheta(i)}\sigma' = R^{\vartheta(i)}f(\sigma{\downarrow_{R,Ax}}) \\
\Leftrightarrow & (f(tr), f(th), \sigma', \vartheta) \vDash \mathit{honest}(i, R)
\end{array}
$$

– $\phi \equiv \mathit{steps}(i, s(m))$ or $\phi \equiv \neg\mathit{steps}(i, s(m))$, where $s \in \{\mathsf{snd}, \mathsf{rcv}\}$. We have

$$
\begin{array}{ll}
& (tr, th, \sigma{\downarrow_{R,Ax}}, \vartheta) \vDash \mathit{steps}(i, s(m)) \\
\Leftrightarrow & (\vartheta(i), s(m^{\#\vartheta(i)})) \in tr \\
\Leftrightarrow & (\vartheta(i), s(f(m)^{\#\vartheta(i)})) \in f(tr) & \text{justified below} \\
\Leftrightarrow & (f(tr), f(th), \sigma', \vartheta) \vDash \mathit{steps}(i, s(f(m)))
\end{array}
$$

We show the second equivalence. The left-to-right implication holds, since ϕ is safe. For the inverse direction (covering the positive literal $\phi \equiv \mathit{steps}(i, s(m))$), suppose that

$$(\vartheta(i), s(f(m)^{\#\vartheta(i)})) \in f(tr).$$

Then there exists $s(m') \in \mathit{Evt}(\mathcal{M}_P)$ such that $(\vartheta(i), s(m'^{\#\vartheta(i)})) \in tr$ and $f(m') = f(m)$. Since ϕ is safe, this implies $m = m'$ and hence $(\vartheta(i), s(m^{\#\vartheta(i)})) \in tr$.

– $\phi \equiv (i, s(m)) \prec (j, s'(m'))$ or $\phi \equiv \neg((i, s(m)) \prec (j, s'(m')))$, where $s, s' \in \{\mathsf{snd}, \mathsf{rcv}\}$.

$$
\begin{array}{ll}
& (tr, th, \sigma{\downarrow_{R,Ax}}, \vartheta) \vDash (i, s(m)) \prec (j, s'(m')) \\
\Leftrightarrow & (\vartheta(i), s(m^{\#\vartheta(i)})) \prec_{tr} (\vartheta(j), s'(m'^{\#\vartheta(j)})) \\
\Leftrightarrow & (\vartheta(i), s(f(m)^{\#\vartheta(i)})) \prec_{f(tr)} (\vartheta(j), s'(f(m')^{\#\vartheta(j)})) & \text{justified below} \\
\Leftrightarrow & (f(tr), f(th), \sigma', \vartheta) \vDash (i, s(f(m))) \prec (j, s'(f(m')))
\end{array}
$$

We show the second equivalence. Note that, the if-direction immediately follows, since ϕ is safe and f is order-preserving for events.

For the only-if direction (covering the case that $\phi \equiv (i, s(m)) \prec (j, s'(m'))$), suppose $(\vartheta(i), s(f(m)^{\#\vartheta(i)})) \prec_{f(tr)} (\vartheta(j), s'(f(m')^{\#\vartheta(j)}))$. Since f is order-preserving for events, there are $s(u), s'(u') \in Evt(\mathcal{M}_P)$ such that

$$(\vartheta(i), s(u^{\#\vartheta(i)})) \prec_{tr} (\vartheta(j), s'(u'^{\#\vartheta(j)}))$$

with $f(u) = f(m)$ and $f(u') = f(m')$. Since ϕ is safe, we have $u = m$ and $u' = m'$, completing the proof of this direction.

– $\phi \equiv secret(m)$.

$$\begin{array}{lll} & (tr, th, \sigma{\downarrow_{R,Ax}}, \vartheta) \not\models secret(m) & \\ \Leftrightarrow & IK(tr)\sigma{\downarrow_{R,Ax}}, IK_0 \vdash_E m\sigma{\downarrow_{R,Ax}} & \\ \Rightarrow & f(IK(tr))\sigma', IK'_0 \vdash_E f(m)\sigma' & \text{by Proposition 5} \\ \Rightarrow & IK(f(tr))\sigma', IK'_0 \vdash_E f(m)\sigma' & f(IK(tr)) = IK(f(tr))) \\ \Leftrightarrow & (f(tr), f(th), \sigma', \vartheta) \not\models secret(f(m)) & \end{array}$$

The inductive cases are routines. This concludes the proof of the theorem. □

F Syntactic criteria for soundness conditions

Conditions (i) in the definition of R, Ax-closedness (Definition 8) and (iii) in the definition of safe formulas (Definition 10) are hard to check in practice, since they universally quantify over ground terms (the first condition) and well-typed R,Ax-normal ground substitutions (the second one). We therefore propose syntactic criteria for verifying these conditions.

F.1 Syntactic criterion for R, Ax-closure

Lemma 13. *Suppose for all types $\tau \in subs(\Pi(E_f^+))$ and all $\{s,t\} \in Ax$ such that $\tau{\downarrow} \cap (\Gamma(s){\downarrow} \cup \Gamma(t){\downarrow}) \neq \emptyset$ implies $\Gamma(s) \preccurlyeq \tau$ and $\Gamma(t) \preccurlyeq \tau$. Then F_f is R, Ax-closed.*

Proof. Let $\tau \in \Pi(E_f^+)$ and u, v be R,Ax-normal such that $u =_{Ax} v$, $u : \tau_u$, $v : \tau_v$. Because of symmetry, it is sufficient to show that $\tau_u \preccurlyeq \tau$ implies $\tau_v \preccurlyeq \tau$. We show this by induction on the derivation $u =_{Ax} v$ depending on the last rule that has been applied. Suppose $\tau_u \preccurlyeq \tau$.

- Reflexivity: In this case, we have $u = v$. Hence it is clear that $\tau_v \preccurlyeq \tau$.
- Axiom: In this case, there is a pair $\{s,t\} \in Ax$ and a substitution σ such that $u = s\sigma$ and $v = t\sigma$. Since $\tau_u \preccurlyeq \tau$, we have $\Gamma(s\sigma) \preccurlyeq \tau$. Moreover, we have $\Gamma(s\sigma) \preccurlyeq \Gamma(s)$. Hence, it holds that $\tau{\downarrow} \cap \Gamma(s){\downarrow} \neq \emptyset$. By assumption, we have $\Gamma(t) \preccurlyeq \tau$. We also have $\Gamma(t\sigma) \preccurlyeq \Gamma(t)$. Therefore, we obtain that $\tau_v \preccurlyeq \tau$.
- Congruence: Suppose that $t = g(t_1, \ldots, t_n)$ for some $g \in \Sigma^n$, $n \geq 1$. We have $u = g(u_1, \ldots, u_n)$ and $t_i =_{Ax} u_i$. Since $\tau_t \preccurlyeq \tau$, either $\tau = msg$ or $\tau = g(\tau_1, \ldots, \tau_n)$ and $\Gamma(t_i) \preccurlyeq \tau_i$ for all $i \in \widetilde{n}$. In the first case, i.e., $\tau = msg$, it is obvious that $\tau_v \preccurlyeq \tau$. In the latter case, by induction hypothesis, we know that $\Gamma(u_i) \preccurlyeq \tau_i$ for all $i \in \widetilde{n}$. This implies $\tau_u \preccurlyeq \tau$ as required.
- Transitivity: In this case, there is a term w such that $t =_{Ax} w$ and $w =_{Ax} u$. By induction hypothesis, we have $\Gamma(w) \preccurlyeq \tau$ and $\tau_u \preccurlyeq \tau$ which concludes this case.

This completes the proof of the lemma. □

F.2 Syntactic criterion for injectiveness-like

In this section, we present a syntactic criterion to justify the satisfaction of condition (ii) in Definition 10 for F_f that is R, Ax-closed. More generally, we want to solve the following problem.

Problem 1. Suppose F_f is R, Ax-closed. Let $t, u \in \mathcal{M}^\sharp$ be terms and $\{t, u\} \subseteq udom(F_f)$. Under which conditions it holds that $t\sigma =_E u\sigma$ for all ground substitutions σ that are R,Ax-normal and well-typed whenever $f(t\sigma) =_E f(u\sigma)$?

If t and u do not contain a variable of type msg, we provide the following criterion.

Proposition 6. *Let $t, u \in \mathcal{M}^\sharp$ be terms such that*

(i) $msg \notin \Gamma(vars(t) \cup vars(u))$,
(ii) $\{t, u\} \subseteq udom(F_f)$.

If $f(t) = t$ and $f(u) = u$ then for all ground substitutions σ that are R,Ax-normal and well-typed, we have that $f(t\sigma) =_E f(u\sigma)$ implies $t\sigma =_E u\sigma$.

Proof. Without loss of generality, we can assume that $dom(\sigma) \subseteq vars(t) \cup vars(u)$. Since $msg \notin \Gamma(vars(t) \cup vars(u))$, we derive that $X\sigma$ is an atom for all $X \in dom(\sigma)$. This implies $f(\sigma) = \sigma$. Moreover, by Theorem 1, we have $f(t\sigma) = f(t)f(\sigma)$ and $f(u)f(\sigma)$. From assumption, we derive

$$f(t\sigma) = t\sigma$$
$$f(u\sigma) = u\sigma$$

Since $f(t\sigma) =_E f(u\sigma)$, we obtain that $t\sigma =_E u\sigma$ as required. This completes the proof of the proposition. □

Next, we present a syntactic criterion that can be employed if either t or u contains a variable of type msg. Before establishing the main result, we prove some auxiliary lemmas. For terms t and u, we use $\rho_{[u/t]}$ to denote the the mapping that replaces each term t' with $t' =_{Ax} t$ by u. We explicitly assume that all pairs $\{s, s'\} \in Ax$ satisfy that $|s\sigma| = |s'\sigma|$ for all substitutions σ. With this assumption, we can treat $\rho_{[u/t]}$ as a normal substitution due to the following result.

Proposition 7. *Let t be a ground term. Suppose that for all pairs $\{s, s'\} \in Ax$ and all substitutions σ, we have $|s\sigma| = |s'\sigma|$. Then for all terms u, u' such that $u =_{Ax} t$ and $u' =_{Ax} t$, it holds that $u \notin subs(u') \setminus \{u'\}$.*

Proof. Suppose that it is not the case, i.e., $u \in subs(u')\setminus\{u'\}$. Then since $u =_{Ax} t$ and $u' =_{Ax} t$, we have $u =_{Ax} u'$. By assumption, we derive that $|u| = |u'|$ which is a contradiction since u is a strict subterm of u'. This completes the proof of the proposition. □

Lemma 14. *Let u, t, v be ground terms and a be an atom such that $a \notin subs(u) \cup subs(t)$. Suppose that $u\rho_{[a/v]} =_{Ax} t\rho_{[a/v]}$. Then $u \in dom(\rho_{[a/v]})$ iff $t \in dom(\rho_{[a/v]})$.*

Proof. By symmetry, it is sufficient to show that $u \in dom(\rho_{[a/v]})$ implies $t \in dom(\rho_{[a/v]})$. Suppose that $u \in dom(\rho_{[a/v]})$. Then we have $u\rho_{[a/v]} = a$. Together with $u\rho_{[a/v]} =_{Ax} t\rho_{[a/v]}$ and a is an atom, this implies $t\rho_{[a/v]} = a$. If $t \in dom(\rho_{[a/v]})$ then we are done. Otherwise, since $a \notin subs(t)$, there must be a strict subterm t' of t such that $t' \in dom(\rho_{[a/v]})$. This implies a is a strict subterm of $t\rho_{[a/v]}$ which is a contradiction. □

Lemma 15. *Let u, t, v be ground terms and a be an atom such that $a \notin subs(u) \cup subs(t)$. Then $u\rho_{[a/v]} = t\rho_{[a/v]}$ implies $u =_{Ax} t$.*

Proof. We show this lemma by induction on u.

- If u is an atom then we have $u\rho_{[a/v]} = u$. Since $u\rho_{[a/v]} = t\rho_{[a/v]}$, we have $t\rho_{[a/v]} = u$. Since $a \notin subs(u)$, it must be the case that $t \notin dom(\rho_{[a/v]})$. Thus, we have $t\rho_{[a/v]} = t$. Therefore, we obtain $u = t$ and hence $u =_{Ax} t$.
- If $u = g(u_1, \ldots, u_n)$ for some $g \in \Sigma^n$ then we consider two cases.
  - If $u \in dom(\rho_{[a/v]})$ then by Lemma 14, we have $t \in dom(\rho_{[a/v]})$. It follows that $t\rho_{[a/v]} = a$. Together with the assumption that $a \notin subs(t)$, this yields $t \in dom(\rho_{[a/v]})$. Hence, we have $u =_{Ax} t$ as required.
  - If $u \notin dom(\rho_{[a/v]})$ then by Lemma 14, we also have $t \notin dom(\rho_{[a/v]})$. Since $top(t\rho_{[a/v]}) = g$ and $t \notin dom(\rho_{[a/v]})$, we must have $top(t) = g$. Therefore, we have $t = g(t_1, \ldots, t_n)$. We derive that

    $$u\rho_{[a/v]} = g(u_1\rho_{[a/v]}, \ldots, u_n\rho_{[a/v]})$$
    $$t\rho_{[a/v]} = g(t_1\rho_{[a/v]}, \ldots, t_n\rho_{[a/v]})$$

    Since $u\rho_{[a/v]} = t\rho_{[a/v]}$, we have $u_i\rho_{[a/v]} = t_i\rho_{[a/v]}$ for all $i \in \widetilde{n}$. By induction hypothesis, we know that $u_i =_{Ax} t_i$. Therefore, by the Congruence rule, we conclude that $u =_{Ax} t$ as required.

This completes the proof of the lemma. □

Lemma 16. *Let t, u, v be terms and a be an atom such that t, v are ground and $a \notin subs(t) \cup subs(u)$. Let σ be a ground substitution such that $dom(\sigma) = vars(u)$ and $t\rho_{[a/v]} = u\sigma$. Then there is a ground substitution σ' such that the following holds.*

(i) $dom(\sigma') = vars(u)$,
(ii) $a \notin subs(ran(\sigma'))$,
(iii) $u\sigma' =_{Ax} t$, *and*
(iv) $\sigma'\rho_{[a/v]} = \sigma$.

Proof. We prove this lemma by induction on u.

- If u is an atom then we have $u\sigma = u$ and thus $t\rho_{[a/v]} = u$. Since $a \notin subs(u)$, we must have $t\rho_{[a/v]} = t = u$. Thus we set σ' to the empty substitution and obtain $u\sigma' = t$. Moreover, we also have $dom(\sigma') = \emptyset = dom(\sigma)$. Hence, it is clear that $\sigma'\rho_{[a/v]} = \sigma$.
- If $u = X$ is a variable then let σ' be such that $dom(\sigma') = \{X\}$ and $X\sigma' = t$. Then we have $u\sigma' = t$. By assumption, we have $a \notin subs(X\sigma) = subs(ran(\sigma))$. Moreover, since $t\rho_{[a/v]} = X\sigma$, it follows that $(X\sigma')\rho_{[a/v]} = X\sigma$. Thus, we also have $\sigma'\rho_{[a/v]} = \sigma$.
- If $u = g(u_1, \ldots, u_n)$ for some $g \in \Sigma^n$ then since $t\rho_{[a/v]} = u\sigma$ and a is an atom, there must be terms $t_1, \ldots, t_n$ such that

  $$t = g(t_1, \ldots, t_n)$$
  $$t\rho_{[a/v]} = g(t_1\rho_{[a/v]}, \ldots, t_n\rho_{[a/v]})$$

Therefore, we have $t_i\rho_{[a/v]} = u_i\sigma$ for all $i \in \widetilde{n}$. By induction hypothesis, there are ground substitutions $\sigma_1, \ldots, \sigma_n$ such that for all $i \in \widetilde{n}$, we have

$$\begin{array}{l} a \notin \mathit{subs}(\mathit{ran}(\sigma_i)), \\ \mathit{dom}(\sigma_i) = \mathit{vars}(u_i), \\ \sigma_i\rho_{[a/v]} = \sigma|_{\mathit{vars}(u_i)}, \text{ and} \\ u_i\sigma_i =_{Ax} t_i. \end{array}$$

We define σ' such that $\mathit{dom}(\sigma') = \mathit{vars}(u)$ and for all $X \in \mathit{dom}(\sigma')$, $X\sigma' = X\sigma_i$ where $i \in \widetilde{n}$ is the smallest index such that $X \in \mathit{dom}(\sigma_i)$. It is clear that $\sigma'\rho_{[a/v]} = \sigma$ and $a \notin \mathit{subs}(\mathit{ran}(\sigma'))$. To see that $u\sigma' =_{Ax} t$, it is sufficient to show that for all $i, j \in \widetilde{n}$ and all $X \in \mathit{dom}(\sigma_i) \cap \mathit{dom}(\sigma_j)$, it holds that $X\sigma_i =_{Ax} X\sigma_j$. Let $i, j \in \widetilde{n}$ and $X \in \mathit{dom}(\sigma_i) \cap \mathit{dom}(\sigma_j)$. From the induction hypothesis, we know that $(X\sigma_i)\rho_{[a/v]} = X\sigma$ and $(X\sigma_j)\rho_{[a/v]} = X\sigma$. Thus, we have $(X\sigma_i)\rho_{[a/v]} = (X\sigma_j)\rho_{[a/v]}$. This by Lemma 15 implies that $X\sigma_i =_{Ax} X\sigma_j$. We therefore conclude this case.

□

Lemma 17. *Let t, u, v be ground terms and a be an atom such that t, v are ground and $a \notin \mathit{subs}(t) \cup \mathit{subs}(u)$. Suppose that*

(i) $t\rho_{[a/v]} =_{Ax} u$, and
(ii) for all $\{s, s'\} \in Ax$, we have $\mathit{top}(v) \notin \mathit{ct}(s) \cup \mathit{ct}(s')$.

Then there is a ground term t' such that $t =_{Ax} t'$ and $t'\rho_{[a/v]} = u$.

Proof. We prove this lemma by induction on the derivation of $t\rho_{[a/v]} =_{Ax} u$ depending on the last rule that has been applied.

- Reflexivity: We have $t\rho_{[a/v]} = u$. Thus, we can pick $t' = t$.
- Axiom: In this case, there are $\{s, s'\} \in Ax$ and a ground substitution σ such that $t\rho_{[a/v]} = s\sigma$ and $u = s'\sigma$. By Lemma 16, there exists a ground substitutions σ' such that

$$\begin{array}{l} \mathit{dom}(\sigma') = \mathit{vars}(s), \\ a \notin \mathit{subs}(\mathit{ran}(\sigma')), \\ s\sigma' =_{Ax} t, \text{ and} \\ \sigma'\rho_{[a/v]} = \sigma. \end{array}$$

  We pick $t' = s'\sigma'$. Since $\mathit{vars}(s) = \mathit{vars}(s')$, we have $s'\sigma'$ is ground. Moreover, we have $t' =_{Ax} t$. By assumption (ii), we derive that $s'\sigma = s'(\sigma'\rho_{[a/v]}) = (s'\sigma')\rho_{[a/v]} = t'\rho_{[a/v]}$. Hence, we have $u = t'\rho_{[a/v]}$ as required.
- Congruence: In this case, there are $c \in \Sigma^n$ and terms $t_1, \ldots, t_n$, $u_1, \ldots, u_n$ such that

$$\begin{array}{l} t = c(t_1, \ldots, t_n), \\ u = c(u_1, \ldots, u_n), \\ t_i\rho_{[a/v]} =_{Ax} u_i \text{ for all } i \in \widetilde{n}. \end{array}$$

  By induction hypothesis, there is a term t'_i such that $t_i =_{Ax} t'_i$ and $u_i = t'_i\rho_{[a/v]}$ for all $i \in \widetilde{n}$. We define $t' = c(t'_1, \ldots, t'_n)$ and derive that $t' =_{Ax} t$ and $u = t'\rho_{[a/v]}$ as required.

– Transitivity: In this case, there is a term w such that $t\rho_{[a/v]} =_{Ax} w$ and $w =_{Ax} u$. By induction hypothesis, there is a term w' such that $t =_{Ax} w'$ and $w = w'\rho_{[a/v]}$. Thus, we have $w'\rho_{[a/v]} =_{Ax} u$. Using the induction hypothesis, we derive that there exists a term t' such that $w' =_{Ax} t'$ and $u = t'\rho_{[a/v]}$. It follows that $t =_{Ax} t'$ which concludes this case.

□

Lemma 18. *Let u, t, v be ground terms and a be an atom. Suppose that for all pairs $\{s, s'\} \in Ax$, we have $top(v) \notin ct(s) \cup ct(s')$. Then $u =_{Ax} t$ implies $u\rho_{[a/v]} =_{Ax} t\rho_{[a/v]}$.*

Proof. We prove the direction from left to right by induction on the derivation $u =_{Ax} t$ depending on the last rule that has been applied.

– Reflexivity: In this case, we have $u = t$. Thus it is obvious that $u\rho_{[a/v]} = t\rho_{[a/v]}$.
– Axiom: In this case, then there are a pair $\{s, s'\} \in Ax$ and a substitution σ such that $u = s\sigma$ and $t = s'\sigma$. Let $\sigma' = \sigma\rho_{[a/v]}$. Since $top(v) \notin ct(s) \cup ct(s')$, we have

$$(s\sigma)\rho_{[a/v]} = s\sigma'$$
$$(s'\sigma)\rho_{[a/v]} = s'\sigma'$$

Therefore, we derive

$$u\rho_{[a/v]} = (s\sigma)\rho_{[a/v]} = s\sigma'$$
$$t\rho_{[a/v]} = (s'\sigma)\rho_{[a/v]} = s'\sigma'$$

Using the Axiom rule, we derive that $u\rho_{[a/v]} =_{Ax} t\rho_{[a/v]}$ as required.
– Congruence: In this case, we have that $u = g(u_1, \ldots, u_n)$ and $t = g(t_1, \ldots, t_n)$ for some $g \in \Sigma^n$, $n \geq 1$. Moreover, we have $u_i =_{Ax} t_i$ for all $i \in \widetilde{n}$. Since $u =_{Ax} t$, it is clear that $u \in dom(\rho_{[a/v]})$ iff $t \in dom(\rho_{[a/v]})$. We consider two cases.
  • If $u \in dom(\rho_{[a/v]})$ then $t \in dom(\rho_{[a/v]})$. Thus, we have $u\rho_{[a/v]} = a = t\rho_{[a/v]}$.
  • If $u \notin dom(\rho_{[a/v]})$ then $t \notin dom(\rho_{[a/v]})$. Therefore, we have

$$u\rho_{[a/v]} = g(u_1\rho_{[a/v]}, \ldots, u_n\rho_{[a/v]})$$
$$t\rho_{[a/v]} = g(t_1\rho_{[a/v]}, \ldots, t_n\rho_{[a/v]})$$

Moreover, by induction hypothesis, we know that $u_i\rho_{[a/v]} =_{Ax} t_i\rho_{[a/v]}$ for all $i \in \widetilde{n}$. Hence, we obtain $u\rho_{[a/v]} =_{Ax} t\rho_{[a/v]}$ as required.
– Transitivity: In this case, there is a term w such that $u =_{Ax} w$ and $w =_{Ax} t$. By induction hypothesis, we have

$$u\rho_{[a/v]} =_{Ax} w\rho_{[a/v]}$$
$$w\rho_{[a/v]} =_{Ax} t\rho_{[a/v]}$$

It follows that $u\rho_{[a/v]} =_{Ax} t\rho_{[a/v]}$ as required.

This completes the proof of the lemma. □

Lemma 19. *Let u, t, v be ground terms and a be an atom such that $a \notin subs(u) \cup subs(t)$. Suppose that for all pairs $\{s, s'\} \in Ax$, we have*

(i) $top(v) \notin ct(s) \cup ct(s')$,
(ii) $a \notin subs(s) \cup subs(s')$, *and*
(iii) $vars(s) = vars(s')$.

Then $u\rho_{[a/v]} =_{Ax} t\rho_{[a/v]}$ implies $u =_{Ax} t$.

Proof. We prove this lemma by induction on the derivation $u\rho_{[a/v]} =_{Ax} t\rho_{[a/v]}$.

- Reflexivity: In this case, we have $u\rho_{[a/v]} = t\rho_{[a/v]}$. By Lemma 15, we have $u =_{Ax} t$ as required.
- Axiom: In this case, there are a pair $\{s, s'\} \in Ax$ and a ground substitution σ such that $dom(\sigma) = vars(s) \cup vars(s')$ and

$$\begin{aligned} u\rho_{[a/v]} &= s\sigma, \\ t\rho_{[a/v]} &= s'\sigma. \end{aligned}$$

  Moreover, by Lemma 16, there is a ground substitution σ' such that $dom(\sigma') = vars(s)$ and

$$\begin{aligned} u &=_{Ax} s\sigma', \\ \sigma'\rho_{[a/v]} &= \sigma|_{vars(s)}\ . \end{aligned}$$

  By assumption (iii), we derive that $\sigma|_{vars(s)} = \sigma$. Hence, we have

$$s'\sigma = s'(\sigma'\rho_{[a/v]})$$

  By assumption (i), for all terms $w \in dom(\rho_{[a/v]}) \cap subs(s'\sigma)$, we have $w \in subs(ran(\sigma))$. Therefore, we must have $s'\sigma = (s'\sigma')\rho_{[a/v]}$. Hence, we obtain

$$t\rho_{[a/v]} = (s'\sigma')\rho_{[a/v]}.$$

  By Lemma 15, we derive that $t =_{Ax} s'\sigma'$. Together with $u =_{Ax} s\sigma'$, we have $u =_{Ax} t$ as required.
- Congruence: In this case, there is $g \in \Sigma^n$ such that

$$\begin{aligned} u\rho_{[a/v]} &= g(u_1, \ldots, u_n) \\ t\rho_{[a/v]} &= g(t_1, \ldots, t_n) \end{aligned}$$

  Moreover, we have $u_i =_{Ax} t_i$ for all $i \in \widetilde{n}$. We consider two cases.
  - If $u \in dom(\rho_{[a/v]})$ then by Lemma 14, we have $t \in dom(\rho_{[a/v]})$. Therefore, we have $u =_{Ax} t$ as required.
  - If $u \notin dom(\rho_{[a/v]})$ then by Lemma 14, we also have $t \notin dom(\rho_{[a/v]})$. Thus, there must be terms $u'_1, \ldots, u'_n$ and terms $t'_1, \ldots, t'_n$ such that

$$\begin{aligned} u &= g(u'_1, \ldots, u'_n), \\ t &= g(t'_1, \ldots, t'_n), \\ u'_i\rho_{[a/v]} &= u_i \text{ for all } i \in \widetilde{n}, \\ t'_i\rho_{[a/v]} &= t_i \text{ for all } i \in \widetilde{n}. \end{aligned}$$

    Hence, we have $u'_i\rho_{[a/v]} =_{Ax} t'_i\rho_{[a/v]}$ for all $i \in \widetilde{n}$. By induction hypothesis, we know that $u'_i =_{Ax} t'_i$ for all $i \in \widetilde{n}$. This implies $u =_{Ax} t$ as required.

– Transitivity: In this case, there is a term w such that $u\rho_{[a/v]} =_{Ax} w$ and $w =_{Ax} t\rho_{[a/v]}$. We consider two cases.
  - If $a \in subs(u\rho_{[a/v]}) \cup subs(t\rho_{[a/v]})$ then from assumption (ii), we derive that $a \in subs(w)$. Hence, there is a term w' such that $w = w'\rho_{[a/v]}$. By induction hypothesis, we have $u =_{Ax} w'$ and $w' =_{Ax} t$. This implies $u =_{Ax} t$ as required.
  - If $a \notin subs(u\rho_{[a/v]}) \cup subs(t\rho_{[a/v]})$ then we have $u\rho_{[a/v]} = u$ and $t\rho_{[a/v]} = t$. Thus, we obtain $u =_{Ax} t$ as required.

This completes the proof of the lemma. □

Lemma 20. *Let t, u, v be terms and a be an atom such that t, v are ground and $a \notin subs(t) \cup subs(u)$. Let σ be a ground substitution such that $dom(\sigma) = vars(u)$ and $t\rho_{[a/v]} =_{Ax} u\sigma$. Suppose that for all pairs $\{s, s'\} \in Ax$, we have*

(i) $top(v) \notin ct(s) \cup ct(s')$,
(ii) $a \notin subs(s) \cup subs(s')$, *and*
(iii) $vars(s) = vars(s')$.

Then there is a ground substitution σ' such that the following holds.

(i) $dom(\sigma') = vars(u)$,
(ii) $a \notin subs(ran(\sigma'))$,
(iii) $u\sigma' =_{Ax} t$, *and*
(iv) $\sigma'\rho_{[a/v]} = \sigma$.

Proof. We prove this lemma by induction on the derivation $t\rho_{[a/v]} =_{Ax} u\sigma$.

– Reflexivity: We have $t\rho_{[a/v]} = u\sigma$ and thus the conclusion follows immediately from Lemma 16.
– Axiom: Suppose that there are a pair $\{s, s'\} \in Ax$ and a ground substitution θ such that $dom(\theta) = dom(s) \cup dom(s')$ and

$$\begin{aligned} t\rho_{[a/v]} &= s\theta \\ u\sigma &= s'\theta \end{aligned}$$

By Lemma 16, there is a ground substitution σ'' such that

$$\begin{aligned} &dom(\sigma'') = vars(s), \\ &a \notin subs(ran(\sigma'')), \\ &\sigma''\rho_{[a/v]} = \theta|_{vars(s)}, \text{ and} \\ &t =_{Ax} s\sigma''. \end{aligned}$$

By assumption (iii), we have $\theta\,|_{vars(s)} = \theta$. Hence, we derive that $s'\theta = s'(\sigma''\rho_{[a/v]})$. This by assumption (i) implies that $s'\theta = (s'\sigma'')\rho_{[a/v]}$. Since $a \notin subs(ran(\sigma''))$, from assumption (ii), we have $a \notin subs(s'\sigma'')$. Moreover, we have $u\sigma = s'\theta = (s'\sigma'')\rho_{[a/v]}$. By Lemma 16, there is a ground substitution σ' such that

$$\begin{aligned} &dom(\sigma') = vars(u), \\ &a \notin subs(ran(\sigma')), \\ &\sigma'\rho_{[a/v]} = \sigma, \text{ and} \\ &s'\sigma'' =_{Ax} u\sigma'. \end{aligned}$$

Together with $t =_{Ax} s\sigma''$, we derive that $u\sigma' =_{Ax} t$.

– Congruence: We have $t = c(t_1, \ldots, t_n)$ and $u = c(u_1, \ldots, u_n)$ for $c \in \Sigma^n$. Moreover, we have $t_i\rho_{[a/v]} =_{Ax} u_i\sigma$ for all $i \in \widetilde{n}$. By induction hypothesis, there are ground substitutions σ_i for all $i \in \widetilde{n}$ such that

$$\begin{array}{l} dom(\sigma_i) = vars(u_i), \\ a \notin subs(ran(\sigma_i)), \\ u_i\sigma_i =_{Ax} t_i, \text{ and} \\ \sigma_i\rho_{[a/v]} = \sigma. \end{array}$$

We can pick $\sigma' = \bigcup_{i=1}^{n} \sigma_i$ that satisfies the desired properties.

– Transitivity: In this case, there is a term w such that $t\rho_{[a/v]} =_{Ax} w$ and $w =_{Ax} u\sigma$. By Lemma 17, there is a ground term t' such that $t =_{Ax} t'$ and $w = t'\rho_{[a/v]}$. Hence, we have $t'\rho_{[a/v]} =_{Ax} u\sigma$. By induction hypothesis, there exists a ground substitution σ' such that

$$\begin{array}{l} dom(\sigma') = vars(u), \\ a \notin subs(ran(\sigma')), \\ u\sigma_i =_{Ax} t', \text{ and} \\ \sigma'\rho_{[a/v]} = \sigma. \end{array}$$

Since $t' =_{Ax} t$, we derive that σ' satisfies the desired properties.

This completes the proof of the lemma. □

Lemma 21. *Let t, v be ground terms and a be an atom such that $a \notin subs(t)$. Suppose that for all pairs $\{s, s'\} \in Ax$, and all rules $l \to r \in R$, we have*

(i) t is R,Ax-normal,
(ii) $a \notin subs(s) \cup subs(s') \cup subs(l)$,
(iii) $top(v) \notin ct(s) \cup ct(s')$, and
(iv) $vars(s) = vars(s')$.

Then $t\rho_{[a/v]}$ is R,Ax-normal.

Proof. We prove this lemma by induction on t.

– If t is an atom then we have $t\rho_{[a/v]} = t$ and thus $t\rho_{[a/v]}$ is R,Ax-normal.
– If $t = g(t_1, \ldots, t_n)$ for some $g \in \Sigma^n$ then we consider two cases.
  • If $t \in dom(\rho_{[a/v]})$ then we have $t\rho_{[a/v]} = a$ and thus $t\rho_{[a/v]}$ is R,Ax-normal.
  • If $t \notin dom(\rho_{[a/v]})$ then we have

$$t\rho_{[a/v]} = g(t_1\rho_{[a/v]}, \ldots, t_n\rho_{[a/v]})$$

Since t is R,Ax-normal, so is t_i for all $i \in \widetilde{n}$. By induction hypothesis, we have $t_i\rho_{[a/v]}$ is R,Ax-normal for all $i \in \widetilde{n}$. There are two smaller cases.
    ∗ If there is no rule $l \to r \in R$ that is applicable to $t\rho_{[a/v]}$ at the root, then since $t_i\rho_{[a/v]}$ is R,Ax-normal for all $i \in \widetilde{n}$, we derive that $t\rho_{[a/v]}$ is R,Ax-normal.

* If there is a rule $l \to r \in R$ that is applicable to $t\rho_{[a/v]}$ at the root, then there is a ground substitution σ such that $dom(\sigma) = vars(l)$ and $t\rho_{[a/v]} =_{Ax} l\sigma$. By Lemma 20, there is a ground substitution σ' such that $t =_{Ax} l\sigma'$. Thus means t is not R,Ax-normal which is a contradiction.

This completes the proof of the lemma. □

Lemma 22. *Let t, v be ground terms such that v is R,Ax-normal and a be an atom such that $a \notin subs(t)$. Suppose that for all pairs $\{s, s'\} \in Ax$, and all rules $l \to r \in R$, we have*

(i) $a \notin subs(s) \cup subs(s') \cup subs(l)$,
(ii) $top(v) \notin ct(s) \cup ct(s')$,
(iii) $vars(s) = vars(s')$, *and*
(iv) there is no non-variable term $u \in subs(l) \setminus \{l\}$ *such that* $top(u) = top(v)$.

Then $(t\rho_{[a/v]})\!\downarrow_{R,Ax} =_{Ax} t\!\downarrow_{R,Ax} \rho_{[a/v]}$.

Proof. We prove this lemma by induction on the length ℓ of the derivation $t \to_{R,Ax} t_1 \to_{R,Ax} \cdots \to_{R,Ax} t_{\ell-1} \to_{R,Ax} t\!\downarrow_{R,Ax}$.

– If $\ell = 0$ then t is R,Ax-normal. By Lemma 21, $t\rho_{[a/v]}$ is R,Ax-normal. Therefore, we have $t\rho_{[a/v]} =_{Ax} (t\rho_{[a/v]})\!\downarrow_{R,Ax}$. Moreover, by Lemma 18, we have $t\rho_{[a/v]} =_{Ax} t\!\downarrow_{R,Ax} \rho_{[a/v]}$. Hence $(t\rho_{[a/v]})\!\downarrow_{R,Ax} =_{Ax} t\!\downarrow_{R,Ax} \rho_{[a/v]}$.
– If $\ell > 0$ then there are a position k, a rule $l \to r \in R$, and a substitution σ such that $t|_k =_{Ax} l\sigma$ and $(t[r\sigma]_k)\!\downarrow_{R,Ax} =_{Ax} t\!\downarrow_{R,Ax}$. By Lemma 18, we have $t|_k\ \rho_{[a/v]} =_{Ax} (l\sigma)\rho_{[a/v]}$. Let $\sigma' = \sigma\rho_{[a/v]}$.
Note that for all terms $u \in subs(t)$ such that $t|_k \in subs(u)$, we must have

$$u \notin dom(\rho_{[a/v]}). \tag{22}$$

Suppose that it is not the case, i.e., we have $u =_{Ax} v$. Then since $t|_k \in subs(u)$ and $t|_k \to_{R,Ax} r\sigma$, we derive that v is not R,Ax-normal which is a contradiction. Therefore, we have established (22). In particular, we have $l\sigma \notin dom(\rho_{[a/v]})$. Moreover, there must not exist a non-variable term $u \in subs(l) \setminus \{l\}$ such that $u\sigma =_{Ax} v$. Because otherwise, by point (ii) in Definition 1, $u\sigma =_{Ax} v$ implies $top(u\sigma) = top(v)$. This means $top(u) = top(v)$ which contradicts assumption (iv). Therefore, we have $(l\sigma)\rho_{[a/v]} = l(\sigma\rho_{[a/v]}) = l\sigma'$. It follows that $t|_k\ \rho_{[a/v]} =_{Ax} l\sigma'$. Hence, we obtain $t[t|_k\ \rho_{[a/v]}]_k =_{Ax} t[l\sigma']_k$. This yields the following.

$$\begin{aligned} t\rho_{[a/v]} &= (t[t|_k\ \rho_{[a/v]}]_k)\rho_{[a/v]} \\ &=_{Ax} (t[l\sigma']_k)\rho_{[a/v]} \qquad \text{by Lemma 18} \end{aligned}$$

Together with (22), we derive that $(t[l\sigma']_k)\rho_{[a/v]} \to_{R,Ax} (t[r\sigma']_k)\rho_{[a/v]}$. Thus, we have $(t\rho_{[a/v]})\!\downarrow_{R,Ax} =_{Ax} ((t[r\sigma']_k)\rho_{[a/v]})\!\downarrow_{R,Ax}$. Note that from assumption (iv) and the fact that $r \in subs(l)$, we derive that $r\sigma' = r(\sigma\rho_{[a/v]}) = (r\sigma)\rho_{[a/v]}$. Therefore, we have

$$\begin{aligned} (t[r\sigma']_k)\rho_{[a/v]} &= (t[(r\sigma)\rho_{[a/v]}]_k)\rho_{[a/v]} \\ &= (t[r\sigma]_k)\rho_{[a/v]} \end{aligned}$$

Hence we have

$$(t\rho_{[a/v]})\!\downarrow_{R,Ax} =_{Ax} ((t[r\sigma]_k)\rho_{[a/v]})\!\downarrow_{R,Ax} \tag{23}$$

Since $(t[r\sigma]_k)\!\downarrow_{R,Ax} =_{Ax} t\!\downarrow_{R,Ax}$, by Lemma 18, we have

$$(t[r\sigma]_k)\!\downarrow_{R,Ax} \rho_{[a/v]} =_{Ax} t\!\downarrow_{R,Ax} \rho_{[a/v]} \tag{24}$$

By induction hypothesis, we have

$$((t[r\sigma]_k)\rho_{[a/v]})\!\downarrow_{R,Ax} =_{Ax} (t[r\sigma]_k)\!\downarrow_{R,Ax} \rho_{[a/v]}$$

This by (23) and (24) yields $(t\rho_{[a/v]})\!\downarrow_{R,Ax} =_{Ax} t\!\downarrow_{R,Ax} \rho_{[a/v]}$ as required.

This completes the proof of the lemma. □

Lemma 23. *Let $T \cup \{t, v\}$ be a set of ground terms such that v is R,Ax-normal and a be a constant such that $a \notin subs(t)$. Suppose that for all pairs $\{s, s'\} \in Ax$, and all rules $l \to r \in R$, we have*

(i) $a \notin subs(s) \cup subs(s') \cup subs(l)$,
(ii) $top(v) \notin ct(s) \cup ct(s')$,
(iii) $vars(s) = vars(s')$, *and*
(iv) there is no non-variable term $u \in subs(l) \setminus \{l\}$ *such that* $top(u) = top(v)$.

Then $T \vdash_E t$ implies $T\rho_{[a/v]}, IK_0 \vdash_E t\rho_{[a/v]}$.

Proof. We prove this lemma induction on the derivation of $T \vdash_E t$ depending on the last rule that has been applied.

- Ax: We have $t \in T$ and thus $t\rho_{[a/v]} \in T\rho_{[a/v]}$. Therefore $T\rho_{[a/v]}, IK_0 \vdash_E t\rho_{[a/v]}$.
- Comp: We have $t = g(u_1, \ldots, t_n)$ and $T \vdash_E u_i$ for $i \in \widetilde{n}$. There are two cases.
  - If $t =_{Ax} v$ then we have $t\rho_{[a/v]} = a$. Since $a \in IK_0$, we obtain that $T\rho_{[a/v]}, IK_0 \vdash_E t\rho_{[a/v]}$.
  - If $t \neq_{Ax} v$ then we have

    $$t\rho_{[a/v]} = g(u_1\rho_{[a/v]}, \ldots, u_n\rho_{[a/v]})$$

    Moreover, by induction hypothesis, we have $T\rho_{[a/v]}, IK_0 \vdash_E u_i\rho_{[a/v]}$ for all $i \in \widetilde{n}$. Hence, we obtain $T\rho_{[a/v]}, IK_0 \vdash_E t\rho_{[a/v]}$ as required.
- Eq: In this case, there is a term t' such that $T \vdash_E t'$ and $t' =_E t$. Since $a \notin subs(t)$, from assumption (i), we derive that $a \notin subs(t')$. Hence, we can apply the induction hypothesis and obtain $T\rho_{[a/v]} \vdash_E t'\rho_{[a/v]}$. To see that $T\rho_{[a/v]} \vdash_E t\rho_{[a/v]}$, it is sufficient to show that $t'\rho_{[a/v]} =_E t\rho_{[a/v]}$. Indeed, from $t' =_E t$, we derive that $t'\!\downarrow_{R,Ax} =_{Ax} t\!\downarrow_{R,Ax}$. This by Lemma 18 implies

  $$t'\!\downarrow_{R,Ax} \rho_{[a/v]} =_{Ax} t\!\downarrow_{R,Ax} \rho_{[a/v]}$$

  By Lemma 22, we have $(t'\rho_{[a/v]})\!\downarrow_{R,Ax} =_{Ax} (t\rho_{[a/v]})\!\downarrow_{R,Ax}$. Therefore, we derive $t'\rho_{[a/v]} =_E t\rho_{[a/v]}$ as desired.

This completes the proof of the lemma. □

In order to establish our syntactic criterion, we further restrict R,Ax-normal attacks to *well-formed* ones.

Definition 11. *A substitution σ is* well-formed *with respect to f if for all $X \in dom(\sigma)$, we have $X\sigma$ and $f(X\sigma)$ are R,Ax-normal.*

We first prove some auxiliary lemmas.

Lemma 24. *Let t, v be ground term, and a be an atom. Assume that*

(i) t is R,Ax-normal,
(ii) v is composed and not a pair,
(iii) for all $\tau \in \Pi(E_f^+)$ and all types $\tau' \in subs(\tau) \setminus \{\tau'\}$, τ' is composed implies $top(\tau') \neq top(v)$, and $\Gamma(a) \preccurlyeq \tau'$ implies $\tau' = msg$,
(iv) for all terms $u \in Rec(F_f, t)$, $u \neq_{Ax} v$,
(v) for all $\{s, s'\} \in Ax$ and $l \to r \in R$, we have
  (a) $a \notin subs(s) \cup subs(s') \cup subs(l)$,
  (b) $top(v) \notin ct(s) \cup ct(s')$, and
  (c) $vars(s) = vars(s')$.

Then $t\rho_{[a/v]}$ is R,Ax-normal and $f(t) = f(t\rho_{[a/v]})$.

Proof. The first conjunct, namely $t\rho_{[a/v]}$ is R,Ax-normal, follows from Lemma 21. We now show that $f(t) = f(t\rho_{[a/v]})$ by induction on the size of t.

- If t is an atom then since v is composed, we have $t\rho_{[a/v]} = t$. Thus we obtain $f(t) = f(t\rho_{[a/v]})$.
- If $t = c(t_1, \ldots, t_n)$ for some $c \in \Sigma^n$ and $n \geq 1$, then let $f(p) = q$ be the pattern in E_f^+ that is chosen for t. Let $p : \tau$. Then we have $\Gamma(t) \preccurlyeq \tau$. We show that $\Gamma(t\rho_{[a/v]}) \preccurlyeq \tau$.
  Suppose there is a position k such that $t|_k =_{Ax} v$. It is sufficient to show that $\Gamma(t[a]_k) \preccurlyeq \tau$. Let us consider two cases.
  - If k is a valid position in τ then we have $\Gamma(t|_k) \preccurlyeq \tau|_k$. Since $t|_k =_{Ax} v$ and v is composed, from assumption (v.b), we have that $t|_k$ is composed and $top(t|_k) = top(v)$.
    If $\tau\,|_k \neq msg$ then since $t\,|_k$ is composed and $\Gamma(t\,|_k) \preccurlyeq \tau\,|_k$, we have $top(\tau|_k) = top(t|_k)$. It follows that

$$top(\tau|_k) = top(v) \tag{25}$$

    Note that $t \in Rec(F_f, t)$. By assumption (iv), we have $t \neq_{Ax} v$. Since $t|_k =_{Ax} v$ and $t \neq_{Ax} v$, we know that k is not the root of t. That means $\tau|_k$ is a strict subterm of τ. Hence, by assumption (iii), we derive that $top(\tau|_k) \neq top(v)$ which contradicts (25). Therefore, we must have that $\tau|_k = msg$ and obtain $\Gamma(t[a]_k) \preccurlyeq \tau$ as desired.
  - If k is not a valid position in τ then there must be a position k' above k such that $\tau|_k = msg$. This also yields $\Gamma(t[a]_k) \preccurlyeq \tau$.

Hence, we have shown that $\Gamma(t\rho_{[a/v]}) \preccurlyeq \tau$. Similarly, it holds that whenever a pattern p' matches $t\rho_{[a/v]}$, it also matches t. This implies t and $t\rho_{[a/v]}$ are abstracted under f by the same clause. Let θ and θ' be substitutions such that $t = p\theta$ and $t\rho_{[a/v]} = p\theta'$. We perform a case distinction on p.

- $p = c(p_1, \ldots, p_n)$ for $c \in \Sigma^n \setminus \{\mathsf{C_{Ax}} \cup \mathsf{C_{Key}}\}$. In this case, we have

$$f(c(p_1, \ldots, p_n)) = \langle e_1, \ldots, e_d \rangle$$

for some $d > 0$ and for all $i \in \widetilde{d}$, e_i is one of the following forms:

1. $e_i = c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n}))$ such that

$$\begin{array}{l} \mathit{set}(\overline{q_j}) \subseteq \mathit{split}(p_j) \text{ for all } j \in \widetilde{n},\ c \neq \langle \cdot, \cdot \rangle, \text{ and} \\ c \in \mathsf{C_R} \Rightarrow \overline{q_n} = [p_n] \end{array}$$

2. $e_i = f(q)$ such that $q \in \mathit{split}(p_j)$ for some $j \in \widetilde{n}$.

and it holds that $\forall j \in pp(c).\ \mathit{split}(p_j) \subseteq Q(j)$ where

$$\begin{array}{rl} Q(j) = & \bigcup\{\mathit{set}(\overline{q_j}) \mid c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n})) \in \{e_1, \ldots, e_d\}\} \\ & \bigcup\{q \mid f(q) \in \{e_1, \ldots, e_d\}\} \end{array}$$

Hence, we have

$$\begin{array}{rl} f(t) &= \langle e_1\theta, \ldots, e_d\theta \rangle \\ f(t\rho_{[a/v]}) &= \langle e_1\theta', \ldots, e_d\theta' \rangle \end{array}$$

To see that $f(t\rho_{[a/v]}) = f(t)$, it is sufficient to show that $e_i\theta = e_i\theta'$ for all $i \in \widetilde{d}$. Let $i \in \widetilde{d}$. We consider two cases.

  - $e_i = c(\widehat{f}(\overline{q_1}), \ldots, \widehat{f}(\overline{q_n}))$. To show that $e_i\theta = e_i\theta'$, it is sufficient to show that $\widehat{f}(\overline{q_j}\theta) = \widehat{f}(\overline{q_j}\theta')$ for all $j' \in \widetilde{n}$. Let $j \in \widetilde{n}$. Since v is not a pair and the fact that $(p\theta)\rho_{[a/v]} = p\theta'$, we have $(w\theta)\rho_{[a/v]} = w\theta'$ for all $w \in \mathit{set}(\overline{q_j})$. Moreover, by assumption (iv), we have $w\theta \neq_{Ax} v$. Note that $w\theta \in \mathit{subs}(t) \setminus \{t\}$. Hence, by induction hypothesis, we have $f(w\theta') = f(w\theta)$. This yields $\widehat{f}(\overline{q_j}\theta) = \widehat{f}(\overline{q_j}\theta')$ as desired.
  - $e_i = f(q)$ with $q \in \mathit{split}(p_j)$ for some $j \in \widetilde{n}$. Since $(p\theta)\rho_{[a/v]} = p\theta'$, we derive that $q\theta' = (q\theta)\rho_{[a/v]}$. Moreover, $q \in \mathit{split}(p_j)$ implies that $q\theta \in \mathit{subs}(t) \setminus \{t\}$. By induction hypothesis, we know that $f(q\theta') = f(q\theta)$. Therefore, we obtain $e_i\theta = e_i\theta'$ as required.

- $p = c(p_1, \ldots, p_n)$ for $c \in \Sigma^n \cap (\mathsf{C_{Ax}} \cup \mathsf{C_{Key}})$. In this case, we have

$$\begin{array}{c} f(p) = c(f(p_1), \ldots, f(p_n)) \\ f(t) = c(f(t_1), \ldots, f(t_n)) \\ f(t\rho_{[a/v]}) = c(f(t_1\rho_{[a/v]}), \ldots, f(t_n\rho_{[a/v]})) \end{array}$$

From assumption (iv), we have $t_i \neq_{Ax} v$ for all $i \in \widetilde{n}$. Thus by the induction hypothesis, we have $f(t_i) = f(t_i\rho_{[a/v]})$. Thus we obtain $f(t) = f(t\rho_{[a/v]})$ as required.

This completes the proof of the lemma. □

We show in the following lemma that whenever a protocol admits an attack then there is a *well-formed* attack. We define the set of function symbols that occur on the left-hand side of rewriting rules in R at the top level by

$$\mathsf{D} = \{top(l) \mid l \to r \in R\}.$$

We call a term t is called R,Ax*-stable* if $t\sigma$ is R,Ax-normal whenever σ is R,Ax-normal and well-typed. A set of terms is R,Ax-stable if all its elements are.

Lemma 25. *Let $T \cup \{t, v, a\}$ be terms such that v is ground, a is a constant, and $a \notin subs(t)$. Let σ be a substitution that is ground, R,Ax-normal, and well-typed. Suppose that the following holds.*

(i) $T \cup \{t\} \in udom(F_f)$,
(ii) *for all terms* $u \in subs(T) \cup subs(t)$, *we have* $f(u)$ *is* R,Ax*-stable,*
(iii) $top(v) \notin ct(IK_0)$,
(iv) $f(v)$ *is not* R,Ax*-normal, and*
(v) *for all pairs* $\{s, t\} \in Ax$ *and all rules* $l \to r \in R$, *we have*
  (a) $(ct(s) \cup ct(s')) \cap \mathsf{D} = \emptyset$,
  (b) $a \notin subs(s) \cup subs(s') \cup subs(l)$,
  (c) $top(v) \notin ct(s) \cup ct(s')$,
  (d) $vars(s) = vars(s')$,
  (e) *there is no non-variable term* $u \in subs(l) \setminus \{l\}$ *such that* $top(u) = top(v)$.

Then $T\sigma, IK_0 \vdash_E t\sigma$ *implies* $T(\sigma\rho_{[a/v]}), IK_0 \vdash_E t(\sigma\rho_{[a/v]})$.

Proof. Suppose that $T\sigma, IK_0 \vdash_E t\sigma$. By Lemma 23, we have

$$(T\sigma)\rho_{[a/v]}, IK_0\rho_{[a/v]}, IK_0 \vdash_E (u\sigma)\rho_{[a/v]}$$

Let $T' = subs(T) \cup subs(t)$. We show the following result.

$$\forall u \in T' \setminus vars(T')\ u\sigma \neq_{Ax} v. \tag{26}$$

Suppose that it is not the case, then there is a term $u \in T'$ such that $u\sigma =_{Ax} v$. By Proposition 3, we have $f(u\sigma) =_{Ax} f(v)$. By assumption (i), we can apply Theorem 1 and obtain $f(u\sigma) = f(u)f(\sigma)$. This by assumption (ii) implies that $f(u\sigma)$ is R,Ax-normal. This is a contradiction since $f(u\sigma) =_{Ax} f(v)$ and $f(v)$ is not R,Ax-normal. Hence, we have shown (26). This implies

$$(T\sigma)\rho_{[a/v]} = T(\sigma\rho_{[a/v]})$$
$$(u\sigma)\rho_{[a/v]} = u(\sigma\rho_{[a/v]})$$

Moreover, by assumption (iii), we derive that $IK_0\rho_{[a/v]} = IK_0$. Therefore, we obtain

$$T(\sigma\rho_{[a/v]}), IK_0 \vdash_E u(\sigma\rho_{[a/v]}).$$

This completes the proof of the lemma. □

Lemma 26. *Let $\phi \in \mathcal{L}_P$, and (tr, th, σ) be a reachable state of P such that σ is ground, R,Ax-normal, and well-typed. Suppose that the following holds.*

(i) for all $\tau \in \Pi(E_f^+)$ and all types $\tau' \in subs(\tau) \setminus \{\tau'\}$, τ' is composed implies $top(\tau') \notin \mathsf{D}$, and for all constants a, we have $\Gamma(a) \preccurlyeq \tau'$ implies $\tau' = msg$,
(ii) for all terms $t \in subs(\mathcal{M}_P \cup Sec_\phi \cup EqTerm_\phi)$, we have $f(t)$ is R,Ax-stable,
(iii) $EqTerm_\phi$ is R,Ax-stable,
(iv) $\mathcal{M}_P \cup Sec_\phi \cup EqTerm_\phi \subseteq udom(F_f)$,
(v) $ct(IK_0) \cap \mathsf{D} = \emptyset$, and
(vi) for all pairs $\{s, t\} \in Ax$ and all rules $l \to r \in R$, we have
(a) $a \notin subs(s) \cup subs(s') \cup subs(l)$,
(b) $\mathsf{D} \cap (ct(s) \cup ct(s')) = \emptyset$,
(c) $vars(s) = vars(s')$,
(d) there is no non-variable term $u \in subs(l) \setminus \{l\}$ such that $top(u) \in \mathsf{D}$.

Then there is a substitution σ' that is well-typed and well-formed with respect to f such that the following holds.

(i) (tr, th, σ') is a reachable state in P, and
(ii) if $(tr, th, \sigma) \not\models \phi$ then $(tr, th, \sigma') \not\models \phi$.

Proof. We define the set of terms $Dec(\sigma)$ as follows.

$$Dec(\sigma) = \{t \mid t \in subs(ran(\sigma)) \wedge top(t) \in \mathsf{D} \wedge f(t) \text{ is not } R,Ax\text{-normal}\}.$$

Let $v_0 \in Dec(\sigma)$ and a_0 be a constant that does not occur in ϕ, $ran(\sigma)$, and E. We define $\sigma_0 = \sigma\rho_{[a_0/v_0]}$.

First, we show that σ_0 is well-typed. Note that σ is well-typed by Definition 2. Let $X \in dom(\sigma_0)$ and suppose that $subs(X\sigma) \cap dom(\rho_{[a_0/v_0]}) \neq \emptyset$. Then $X\sigma$ is composed. Since σ is well-typed, we must have $X : msg$. Therefore, we have $\Gamma((X\sigma_0)\downarrow_{R,Ax}) \preccurlyeq \Gamma(X)$. Hence σ_0 is well-typed.

Second, we show that σ_0 is R,Ax-normal. Let $X \in dom(\sigma_0)$. Since σ is R,Ax-normal, so is $X\sigma$. By Lemma 21, we have $(X\sigma)\rho_{[a_0/v_0]}$ is R,Ax-normal. Thus $X\sigma\sigma_0$ is R,Ax-normal. Hence σ_0 is R,Ax-normal.

Third, we show that $|Dec(\sigma_0)| < |Dec(\sigma)|$. For this purpose, it is sufficient to show that for all terms $t \in subs(ran(\sigma))$ such that $f(t)$ is R,Ax-normal, we also have $f(t\rho_{[a_0/v_0]})$ is R,Ax-normal. Let $t \in subs(ran(\sigma))$ such that $f(t)$ is R,Ax-normal. We claim that for all $u \in Rec(F_f, t)$, it holds that $u \neq_{Ax} v_0$. To see that, let us pick an arbitrary term $u \in Rec(F_f, t)$. Since $f(t)$ is R,Ax-normal, so is $f(u)$. Suppose that $u =_{Ax} v_0$. By Proposition 3, we know that $f(u) =_{Ax} f(v_0)$. Since $f(v_0)$ is not R,Ax-normal, neither is $f(u)$. This is a contradiction and therefore, we must have $u \neq_{Ax} v_0$. Hence, we have established that $u \neq_{Ax} v_0$ for all $u \in Rec(F_f, t)$. Thus by Lemma 24, we have $f(t) = f(t\rho_{[a_0/v_0]})$. Since $f(t)$ is R,Ax-normal, so is $f(t\rho_{[a_0/v_0]})$. Hence, we have just proved that

$$|Dec(\sigma_0)| < |Dec(\sigma)|. \tag{27}$$

Fourth, we show reachability for σ_0, i.e., we need to show that (tr, th, σ_0) is reachable in P. We prove this by induction on the number n of transitions leading to (tr, th, σ).

– Base case ($n = 0$): Since (ϵ, th, σ) is reachable, so is (ϵ, th, σ_0).
– Inductive case ($n = k + 1$): Suppose (tr', th', σ) is reachable in k steps and there is a transition $(tr', th', \sigma) \to (tr, th, \sigma)$. By induction hypothesis, we have

$$(tr', th', \sigma) \text{ is reachable in } P. \tag{28}$$

We consider two cases according to the rule r that has been applied in step $k + 1$.

- If $r = SEND$ then it is obvious that $(tr', th', \sigma_0) \to (tr, th, \sigma_0)$. This by (28) yields that (tr, th, σ') is reachable in P.
- If $r = RECV$ then there exists $i \in TID$ and $R \in dom(P)$ such that $th'(i) = (R, \mathsf{rcv}(u) \cdot tl)$, $tr = tr' \cdot (i, \mathsf{rcv}(u))$, $th = th'[i \mapsto (R, tl)]$, and

$$IK(tr')\sigma, IK_0 \vdash_E u\sigma$$

By Lemma 25, we derive that

$$IK(tr')\sigma_0, IK_0 \vdash_E u\sigma_0$$

Thus the reachability of (tr, th, σ_0) in P follows immediately.

Next, we show attack preservation for σ_0, i.e., we need to show that

$$\forall \vartheta.\ (tr, th, \sigma, \vartheta) \not\models \phi \Rightarrow (tr, th, \sigma_0, \vartheta) \not\models \phi$$

We prove this by induction on the structure of ϕ. It is enough to consider the following cases.

– $\phi \equiv m = m'$ or $\phi \equiv \neg(m = m')$.

$$\begin{aligned}
&(tr, th, \sigma, \vartheta) \models m = m' \\
\Leftrightarrow\ & m\sigma =_E m'\sigma \\
\Leftrightarrow\ & m\sigma =_{Ax} m'\sigma && \text{since } \sigma \text{ is } R,Ax\text{-normal, assumption (iii)} \\
\Leftrightarrow\ & (m\sigma)\rho_{[a_0/v_0]} =_{Ax} (m'\sigma)\rho_{[a_0/v_0]} && \text{by Lemmas 18, 15} \\
\Leftrightarrow\ & m(\sigma\rho_{[a_0/v_0]}) =_{Ax} m'(\sigma\rho_{[a_0/v_0]}) && \text{by (26)} \\
\Leftrightarrow\ & m\sigma_0 =_{Ax} m\sigma_0 \\
\Leftrightarrow\ & m\sigma_0 =_E m\sigma_0 && \text{since } \sigma_0 \text{ is } R,Ax\text{-normal, assumption (iii)} \\
\Leftrightarrow\ & (tr, th, \sigma_0, \vartheta) \models m = m'
\end{aligned}$$

– $\phi \equiv honest(i, R)$ or $\phi \equiv \neg honest(i, R)$.

$$\begin{aligned}
&(tr, th, \sigma, \vartheta) \models honest(i, R) \\
\Leftrightarrow\ & R^{\vartheta(i)}\sigma \in \mathcal{A}_H \\
\Leftrightarrow\ & R^{\vartheta(i)}\sigma' \in \mathcal{A}_H && \text{since } R^{\vartheta(i)}\sigma = R^{\vartheta(i)}\sigma_0 \\
\Leftrightarrow\ & (tr, th, \sigma_0, \vartheta) \models honest(i, R)
\end{aligned}$$

– $\phi \equiv secret(m)$.

$$\begin{aligned}
&(tr, th, \sigma, \vartheta) \not\models secret(m) \\
\Leftrightarrow\ & IK(tr)\sigma, IK_0 \vdash_E m\sigma \\
\Rightarrow\ & IK(tr)\sigma_0, IK_0 \vdash_E m\sigma_0 && \text{by Lemma 25} \\
\Leftrightarrow\ & (tr, th, \sigma_0, \vartheta) \not\models secret(m)
\end{aligned}$$

The inductive cases are trivial.

Hence, in this way, we can construct a sequence of substitutions $\sigma_0, \sigma_1, \ldots, \sigma_n$ for $n \geq 0$. By (27), we have $|Dec(\sigma)| > |Dec(\sigma_0)| > \cdots > |Dec(\sigma_n)|$. This sequence is finite and finally we reach some σ_m such that $|Dec(\sigma_m)| = 0$. This means σ_m is well-formed with respect to f. By setting $\sigma' = \sigma_m$, we completes the proof of the lemma. □

Definition 12. *Let $F_f = (f, E_f)$ be a typed abstraction and g be an non-zero arity function symbols. We say that*

- *f is* composite-preserving *if for all $(f(p) = u) \in E_f$ such that $top(p) \in \Sigma^n$ with $n \geq 1$, we have $top(q) \in \Sigma^m \setminus \{f\}$ with $m \geq 1$.*
- *E_f is* constructor-exclusive *for g if for all $(f(p) = q) \in E_f$ we have $top(q) = g$ implies $top(p) = g$.*

Intuitively, E_f is composite-preserving if it never maps a composed term to an atom, and E_f is constructor-exclusive for g if every term with top-level constructor g can only be obtained from one of the same form.

We extend the above notions to a set of symbols $S \subseteq \Sigma$ as expected, i.e, F_f is constructor-exclusive for S if it is for each $g \in S$.

Lemma 27. *Let t be an R,Ax-normal ground term, g be a function symbol, and $F_f = (f, E_f)$ be a typed abstraction such that*

(i) E_f is composite-preserving and constructor-exclusive for g,
(ii) f is homomorphic for g.

Then $top(t) = g$ iff $top(f(t)) = g$.

Proof. We prove by case distinction on the shape of t.

- If t is atomic then we have $f(t) = t$ and thus the lemma holds trivially.
- Now we assume that $t = g'(t_1, \ldots, t_n)$ for some constructor $g' \in \Sigma^n$, $n \geq 1$, and terms $t_1, \ldots, t_n$. Then there exists the first pattern $(f(p) = q) \in E_f^1$ such that $\Gamma(t) \preccurlyeq \Gamma(p)$ and $p\theta = t$ for some substitution θ. Thus, we have $p = g'(p_1, \ldots, p_n)$ for some terms $p_1, \ldots, p_n$. Moreover, we have $f(t) = q\theta$. Since p is composed, we have three cases.
  - If $(f(p) = q) \in E_f^0$ then we know that $top(q) = top(p) = top(t) = g'$. We also have $top(t) = top(q\theta) = top(f(t)) = g'$. Therefore the lemma is trivial.
  - If $(f(p) = q) \in E_f$ then since F_f is composite-preserving, we obtain that $top(t) = top(p)$ and $top(f(t)) = top(q)$. Moreover, since f is homomorphic and constructor-exclusive for g, we conclude this case.

This completes the proof of the lemma. □

We show in the following lemma that, under certain conditions, an equality of abstracted terms implies an equality of the original terms.

Lemma 28. *Let $t, u \in \mathcal{N}$ be R,Ax-normal ground terms and suppose that*

(i) $f(t)$ *and* $f(u)$ *are* R,Ax*-normal,*
(ii) E_f *is composite-preserving and constructor-exclusive for* $ct(t)$*,*
(iii) f *is homomorphic for* $ct(t)$*,*
(iv) there are no pair $\{s, s'\} \in Ax$*, no substitution* θ*, and no term* $u' \in subs(f(u))$ *such that* $u' = s\theta$ *or* $u' = s'\theta$*.*

Then $f(u) =_E f(t)$ *implies* $u = t$*.*

Proof. Suppose that $f(u) =_E f(t)$. We show that $u =_E t$ by induction on $|t|+|u|$.

- If t is an atom then we have $f(t) = t$. Since F_f is composite-preserving, we derive that u is atomic. We also have $f(u) = u$ and thus $u = t$ as required.
- If $t = g(t_1, \ldots, t_n)$ for some $g \in \Sigma^n$ and $n \geq 1$, then by Lemma 27 we have $top(f(t)) = g$. From assumption (i), we derive that $f(t) =_{Ax} f(u)$. This by assumption (iv), we must have $f(t) = f(u)$. This yields $top(f(u)) = g$. Applying Lemma 27 again, we derive that $top(u) = g$, i.e., $u = g(u_1, \ldots, u_n)$ for some terms $u_1, \ldots, u_n$. Moreover, by assumption (iii), we know that f is homomorphic for g. Thus, we derive that $f(t) = g(f(t_1), \ldots, f(t_n))$ and $f(u) = g(f(u_1), \ldots, f(u_n))$. Since $f(t) = f(u)$, by assumption (iv), we derive that $f(t_i) = f(u_i)$ for all $i \in \widetilde{n}$. By induction hypothesis, we have $t_i = u_i$. Therefore, we obtain $t = u$. This completes the proof of the lemma.

□

We now establish a syntactic criterion to justify condition (iii) for equalities in which message variables can only occur in either side. Note that by Lemma 26, it is sufficient to consider the syntactic criterion with respect to well-formed attacks provided that all conditions (i)-(vi) in Lemma 26 are satisfied for a given protocol P and a typed abstraction F_f. We therefore state our main result for well-formed substitutions in the following proposition.

Proposition 8. *Let* $t, u \in \mathcal{M}^\sharp$ *be* R,Ax*-normal terms and suppose that*

(i) $msg \notin \Gamma(vars(t))$ *and* $msg \in \Gamma(vars(u))$*,*
(ii) $\{t, u\} \subseteq udom(F_f)$*,*
(iii) $f(t)$ *and* $f(u)$ *are* R,Ax*-stable,*
(iv) E_f *is composite-preserving and constructor-exclusive for* $ct(t)$*,*
(v) f *is homomorphic for* $ct(t)$*, and*
(vi) there are no pair $\{s, s'\} \in Ax$*, no substitution* θ*, and no term* $t' \in subs(f(t))$ *such that* $t' = s\theta$ *or* $t' = s'\theta$*.*

Then for all ground substitutions σ *that are well-typed and well-formed with respect to* f *such that* $vars(t) \cup vars(u) \subseteq dom(\sigma)$ *and* $f(u\sigma) =_E f(t\sigma)$*, we have* $u\sigma =_E t\sigma$*.*

Proof. Since $msg \notin vars(t)$, all variables in t are of atomic types. Moreover, since σ is well-typed, we have $\Gamma(t\sigma) \preccurlyeq \Gamma(t)$. It follows that $ct(t\sigma) = ct(t)$. Since E_f is constructor-exclusive and f is homomorphic for $ct(t)$, so is it for $ct(t\sigma)$. Moreover, by Theorem 1, we have $f(t\sigma) = f(t)f(\sigma)$ and $f(u\sigma) = f(u)f(\sigma)$. This, together with assumption (iii), implies that $f(t\sigma)$ and $f(u\sigma)$ are R,Ax-normal.

Since $X\sigma$ is an atom for all $X \in dom(\sigma)$, by assumption (vi), we derive that for all $t' \in subs(f(t\sigma))$, there are no pair $\{s, s'\} \in Ax$ and substitution θ such that $t' = s\theta$ or $t' = s'\theta$. Hence, we can apply Lemma 28 and obtain that $u\sigma = t\sigma$ as required. □

F.3 Alternative syntactic criterion

In this section, we discuss a syntactic criterion for condition (iii) in Definition 10 when there are no equations, i.e., $Ax = \emptyset$. We assume that the hypothesis required to apply Lemma 26 holds and restrict the scope of condition (iii) to well-formed substitutions. A violation of this condition exists if the following decision problem has YES answer.

Problem 2. Let $t, u \in \mathcal{T}$ and suppose that

(i) t and u are R,Ax-stable,
(ii) for all $t' \in subs(t) \cup subs(u)$, we have $f(t)$ and $f(u)$ are R,Ax-stable, and
(iii) $t, u \in udom(F_f)$.

Is there a ground substitution σ that is well-typed and well-formed with respect to f such that $f(t\sigma) =_E f(u\sigma)$ and $t\sigma \neq_E u\sigma$.

Since t, u are R,Ax-stable, $Ax = \emptyset$, and σ is well-formed with respect to f, we see that $f(t\sigma) =_E f(u\sigma)$ iff $f(t\sigma) = f(u\sigma)$ and $t\sigma \neq_E u\sigma$ iff $t\sigma \neq u\sigma$. Moreover, by the substitution property, the equality $f(t\sigma) = f(u\sigma)$ is equivalent to $f(t\sigma) = f(u)f(\sigma)$. When u does not contain variables of type msg, we can further reduce the final equality to $f(t\sigma) = f(u)\sigma$. Thus σ can be seen as a solution to the constraints $f(t) = eval_f(f(u))$ and $t \neq u$, where $eval_f(v)$ denotes the explicit evaluation of v for all terms v that contain f.

This observation motivates us to develop a sound and complete constraint-solving algorithm that, if it terminates, either produces a solution or concludes that the constraint system is not satisfiable.

We first define our constraint system.

Constraint system A *constraint* C is either an atom constraint $atom(u)$ or an equality constraint $u \doteq v$ or a disequality constraint $\neg(u \doteq v)$, or an abstraction constraint $t \rhd_f u$, or a type constraint $\mathbf{T}(u) \preccurlyeq \tau$ or $\neg(\mathbf{T}(u) \preccurlyeq \tau)$, where u, v, t are terms and t contains the function symbol f.

We now provide a semantics for our constraints. Let θ be a ground substitution that is well-formed with respect to f. We define θ *satisfies* $\mathcal{C}$ as follows.

$$
\begin{array}{ll}
\theta \Vdash atom(u) & \text{iff } u\theta \text{ is an atom} \\
\theta \Vdash u \doteq v & \text{iff } u\theta = v\theta \\
\theta \Vdash \neg(u \doteq v) & \text{iff } u\theta \neq v\theta \\
\theta \Vdash t \rhd_f u & \text{iff } eval_f(t\theta) = u\theta \\
\theta \Vdash \mathbf{T}(u) \preccurlyeq \tau & \text{iff } \Gamma(u\theta) \preccurlyeq \tau \\
\theta \Vdash \neg(\mathbf{T}(u) \preccurlyeq \tau) & \text{iff } \neg(\Gamma(u\theta) \preccurlyeq \tau)
\end{array}
$$

A *constraint system* $\mathcal{C}$ is a finite set of constraints. We say that θ *satisfies* $\mathcal{C}$, written $\theta \Vdash \mathcal{C}$, if it satisfies each constraint in $\mathcal{C}$. We also call θ a *model* of $\mathcal{C}$. We abuse notation by writing $vars(C)$ to denote the set of variables occurring in the constraint C. We say that a constraint C involves a term t if $vars(t) \cap vars(C) \neq \emptyset$. A constraint system $\mathcal{C}$ involves t of one of its elements does. We define the following sets of constraints.

$$
\begin{aligned}
cs_{\mathsf{deq}}(\mathcal{C}, t) &= \{\neg(u \doteq v) \mid \neg(u \doteq v) \in \mathcal{C} \wedge vars(t) \cap (vars(u) \cup vars(v)) \neq \emptyset\} \\
cs_{\mathsf{type}}(\mathcal{C}, t) &= \{\mathbf{T}(u) \preccurlyeq \tau, \neg(\mathbf{T}(u) \preccurlyeq \tau) \mid \mathbf{T}(u) \preccurlyeq \tau \in \mathcal{C} \wedge vars(t) \cap vars(u) \neq \emptyset\} \\
cs_{\mathsf{atom}}(\mathcal{C}, t) &= \{atom(u) \mid atom(u) \in \mathcal{C} \wedge vars(t) \cap vars(u) \neq \emptyset\}
\end{aligned}
$$

Intuitively, $cs_{\mathsf{deq}}(\mathcal{C}, t)$, $cs_{\mathsf{type}}(\mathcal{C}, t)$, and $cs_{\mathsf{atom}}(\mathcal{C}, t)$ denote the sets of disequality constraints, type constraints, and atom constraints involving t in $\mathcal{C}$, respectively.

Constraint-solving algorithm Our constraint-solving algorithm relies on a set of *constraint reduction rules* (see Figure 3). Intuitively, these rules allow us to reduce a constraint system to a set of simpler ones. A set of constraint systems represents case distinctions. Before presenting these rules in detail, we introduce some definitions. For a given term t, we define the set of constraints $notst(t)$ as

$$notst(t) = \{\neg(\mathbf{T}(t) \preccurlyeq \Gamma(p)) \mid f(p) = q \in E_f \wedge p \notin \mathcal{V}_{pt}\}$$

Intuitively, $notst(t)$ expresses the necessary and sufficient condition that t cannot be abstracted by any clause $f(p) = q \in E_f$ in which p is composed.

We say that a type τ is *singleton* if it has exactly one inhabitant. In this case, we also use $inhb(\tau)$ to refer to this inhabitant of τ.

Our constraint-solving algorithm exploits the constraint reduction relation $\rightsquigarrow$ between constraint systems to sets of constraint systems. We use $\bot$ to denote the unsatisfiable constraint system.

Note that for each clause $(f(p) = q) \in E^1_f$, the clause $f(p^*) = q^*$ denote its renamed version apart from the constraint system on the left-hand side. We also use $\parallel C_1 \parallel \ldots \parallel C_k$ to distinguish k cases. We say that a constraint system is *solved* if no reduction rule is applicable. Our constraint reduction rules simplify a constraint system in four different ways. In particular, each rule either

- removes contradictions, or
- collapses several constraints into a simpler one, or
- splits a complex constraint into several simpler ones, or
- results in a case distinction.

We now explain each group of rules.

Typing reduction rules These rules aim to solve constraints that involve typing. More precisely, rule $R^{dup,el}_{\preccurlyeq}$ collapses two type constraints into an equivalent one. Rules $R^{elim}_{\preccurlyeq}$, $R^{elim}_{\neg,\preccurlyeq}$, and $R^{elim}_{\preccurlyeq}$ remove trivial constraints from the constraint systems. Rule $R^{single}_{\preccurlyeq}$ solves a subtyping, where the type in the constraint is singleton, by adding an equality constraint between the term

Typing reduction rules

$$R^{dup,el}_{\preccurlyeq} : (\mathbf{T}(t) \preccurlyeq \tau, \mathbf{T}(t) \preccurlyeq \tau', \mathcal{C}) \rightsquigarrow (\mathbf{T}(t) \preccurlyeq \tau, \mathcal{C}) \text{ if } \tau \preccurlyeq \tau'$$

$$R^{dup,\bot}_{\preccurlyeq} : (\mathbf{T}(t) \preccurlyeq \tau, \mathbf{T}(t) \preccurlyeq \tau', \mathcal{C}) \rightsquigarrow \bot \quad \text{if } \neg(\tau \preccurlyeq \tau') \wedge \neg(\tau \preccurlyeq \tau')$$

$$R^{\bot}_{\preccurlyeq,\neg,\preccurlyeq} : (\mathbf{T}(t) \preccurlyeq \tau, \neg(\mathbf{T}(t) \preccurlyeq \tau'), \mathcal{C}) \rightsquigarrow \bot \quad \text{if } \tau \preccurlyeq \tau'$$

$$R^{single}_{\preccurlyeq} : (\mathbf{T}(t) \preccurlyeq \tau, \mathcal{C}) \rightsquigarrow (t \doteq \mathit{inhb}(\tau), \mathcal{C}) \quad \text{if } \tau \text{ singleton}$$

$$R^{split}_{\preccurlyeq} : (\mathbf{T}(g(t_1,\ldots,t_k)) \preccurlyeq g(\tau_1,\ldots,\tau_k), \mathcal{C}) \rightsquigarrow (\mathbf{T}(t_1) \preccurlyeq \tau_1, \ldots, \mathbf{T}(t_k) \preccurlyeq \tau_k, \mathcal{C}) \quad \text{if } g \in \Sigma^k, k > 0$$

$$R^{sim}_{\preccurlyeq} : (\mathbf{T}(t) \preccurlyeq g(\tau_1,\ldots,\tau_k), \mathcal{C}) \rightsquigarrow (t \doteq g(X_1,\ldots,X_k), \mathbf{T}(X_1) \preccurlyeq \tau_1, \ldots, \mathbf{T}(X_k) \preccurlyeq \tau_k, \mathcal{C}) \quad \text{if } t \in \mathcal{V}, g \in \Sigma^k, k > 0, X_i s \text{ fresh, pairwise distinct}$$

$$R^{elim}_{\preccurlyeq} : (\mathbf{T}(t) \preccurlyeq \tau, \mathcal{C}) \rightsquigarrow \mathcal{C} \quad \text{if } \tau = \mathit{msg} \text{ or } t \text{ atomic and } \Gamma(t) \preccurlyeq \tau$$

$$R^{\bot}_{\preccurlyeq} : (\mathbf{T}(t) \preccurlyeq \tau, \mathcal{C}) \rightsquigarrow \bot \quad \text{if } t \text{ atomic and } \neg(\Gamma(t) \preccurlyeq \tau) \text{ or } \mathit{atom}(t) \in \mathcal{C} \text{ and } \tau \text{ composed or } t \text{ composed and } \tau \neq \mathit{msg} \text{ atomic or } t, \tau \text{ composed and } \mathit{top}(t) \neq \mathit{top}(\tau)$$

$$R^{split}_{\neg,\preccurlyeq} : (\neg(\mathbf{T}(g(t_1,\ldots,t_k)) \preccurlyeq g(\tau_1,\ldots,\tau_k)), \mathcal{C}) \rightsquigarrow \|_{1\le i\le k} (\neg(\mathbf{T}(t_i) \preccurlyeq \tau_i), \mathcal{C}) \quad \text{if } g \in \Sigma^k, k > 0$$

$$R^{elim}_{\neg,\preccurlyeq} : (\neg(\mathbf{T}(t) \preccurlyeq \tau), \mathcal{C}) \rightsquigarrow \mathcal{C} \quad \text{if } t \text{ atomic and } \neg(\Gamma(t) \preccurlyeq \tau) \text{ or } \mathit{atom}(t) \in \mathcal{C} \text{ and } \tau \text{ composed or } t \text{ composed and } \tau \neq \mathit{msg} \text{ atomic or } t, \tau \text{ composed and } \mathit{top}(t) \neq \mathit{top}(\tau)$$

$$R^{single}_{\preccurlyeq} : (\neg(\mathbf{T}(t) \preccurlyeq \tau), \mathcal{C}) \rightsquigarrow (\neg(t \doteq \mathit{inhb}(\tau)), \mathcal{C}) \quad \text{if } \tau \text{ singleton}$$

$$R^{\bot}_{\neg,\preccurlyeq} : (\neg(\mathbf{T}(t) \preccurlyeq \tau), \mathcal{C}) \rightsquigarrow \bot \quad \text{if } \tau = \mathit{msg} \text{ or } t \text{ atomic and } \Gamma(t) \preccurlyeq \tau$$

Term reduction rules

$$R^{sim}_{\doteq} : (t \doteq u, \mathcal{C}) \rightsquigarrow \mathcal{C}\sigma \quad \text{if } \sigma = \mathit{mgu}(t,u) \text{ exists}$$

$$R^{\bot}_{\doteq} : (t \doteq u, \mathcal{C}) \rightsquigarrow \bot \quad \text{if } \mathit{mgu}(t,u) \text{ does not exist}$$

$$R^{split}_{\neg,\doteq} : (\neg(g(t_1,\ldots,t_k) \doteq g(u_1,\ldots,u_k)), \mathcal{C}) \rightsquigarrow \|_{1\le i\le k} (\neg(t_i \doteq u_i), \mathcal{C}) \quad \text{if } g \in \Sigma^k, k > 0$$

$$R^{elim}_{\neg,\doteq} : (\neg(t \doteq u), \mathcal{C}) \rightsquigarrow \mathcal{C} \quad \text{if } \mathit{top}(t) \neq \mathit{top}(u)$$

$$R^{\bot}_{\neg,\doteq} : (\neg(t \doteq t), \mathcal{C}) \rightsquigarrow \bot$$

$$R^{elim}_{atom} : (\mathit{atom}(u), \mathcal{C}) \rightsquigarrow \mathcal{C} \quad \text{if } u \text{ is an atom}$$

$$R^{\bot}_{atom} : (\mathit{atom}(u), \mathcal{C}) \rightsquigarrow \bot \quad \text{if } u \text{ is composed}$$

Abstraction reduction rules

$$R^{dup}_{\rhd_f} : (f(t) \rhd_f u, f(t) \rhd_f v, \mathcal{C}) \rightsquigarrow (f(t) \rhd_f u, u \doteq v, \mathcal{C})$$

$$R^{split}_{\rhd_f} : (g(t_1,\ldots,t_k) \rhd_f g(u_1,\ldots,u_k), \mathcal{C}) \rightsquigarrow (t_1 \rhd_f u_1, ..., t_k \rhd_f u_k, \mathcal{C}) \quad \text{if } g \neq f$$

$$R^{\bot}_{\rhd_f} : (t \rhd_f u, \mathcal{C}) \rightsquigarrow \bot \quad \text{if } \mathit{top}(t) \neq f \text{ and } \mathit{top}(t) \neq \mathit{top}(u)$$

$$R^{alg}_{\rhd_f} : (f(t) \rhd_f u, \mathcal{C}) \rightsquigarrow \|_{f(p)=q\in E_f, p\notin\mathcal{V}_{pt}} (\mathbf{T}(t) \preccurlyeq \Gamma(p), t \doteq p^*, q^* \rhd_f u, \mathcal{C}) \;\|_{f(p)=q\in E^0_f, p\notin\mathcal{V}_{pt}} (\mathbf{T}(t) \preccurlyeq \Gamma(p), t \doteq p^*, q^* \rhd_f u, \mathit{notst}(t), \mathcal{C}) \;\| (\mathit{atom}(t), t \doteq u, \mathcal{C})$$
if $t \notin \mathcal{V}$ or u composed, or $cs_{\mathsf{deq}}(\mathcal{C}, t) \cup cs_{\mathsf{type}}(\mathcal{C}, t) \neq \emptyset$

Fig. 3. Constraint reduction rules

and the unique inhabitant of the type to the constraint system. Rules $R^{split}_{\preccurlyeq}$ and $R^{split}_{\neg,\preccurlyeq}$ decompose a type constraint into several ones. All the other typing rules derive trivial contradictions basing on the subtyping derivations.

Term reduction rules These rules are responsible to solve constraints for terms. In particular, rules $R^{sim}_{\doteq}$ and $R^{\perp}_{\doteq}$ solves an equality of two terms. Rule $R^{split}_{\neg,\doteq}$ performs a case distinction to solve a disequality. Rules $R^{elim}_{\neg,\doteq}$ and R^{elim}_{atom} remove trivial constraints. Rule $R^{\perp}_{\neg,\doteq}$ derives a contradiction from a disequality constraint. Rule $R^{\perp}_{atom}$ relies on the fact that no instance of a composed term is an atom.

Abstraction rules These rules solves constraints that involve the protocol abstraction f. More concretely, the rule $R^{dup}_{\rhd_f}$ solves two equalities by adding another one to the constraint system. The rule $R^{split}_{\rhd_f}$ splits an equality into several ones by stripping off the outermost constructor from both sides. Rule $R^{\perp}_{\rhd_f}$ derives a contradiction when the left-hand side term's top constructor is not f and disagree with the right-hand side term's. Finally, rule $R^{alg}_{\rhd_f}$ solves an abstraction constraint by performing a case distinction. The side's condition allows us to achieve termination and construct a model.

In the following theorem, we show that the reduction relation $\rightsquigarrow$ is sound and complete.

Theorem 6. *Whenever $\mathcal{C}_1 \rightsquigarrow \mathcal{C}_2$, $\mathcal{C}_1$ has a model iff $\mathcal{C}_2$ does.*

Proof. We first show the direction from left to right. Suppose that θ is a model of $\mathcal{C}_1$ and the side-condition of some rule R is satisfied. Without loss of generality, we can assume that $dom(\theta) \subseteq vars(\mathcal{C}_1)$. We perform a case distinction on R (trivial cases are omitted).

- $R^{single}_{\preccurlyeq}$: From the rule's side condition, we have $\theta \Vdash \mathbf{T}(t) \preccurlyeq \tau$. Therefore $\Gamma(t\theta) \preccurlyeq \tau$. Since τ is singleton, we must have $t\theta = inhb(\tau)$. Thus $\theta \Vdash t \doteq inhb(\tau)$. Hence, we conclude this case.
- $R^{split}_{\preccurlyeq}$: From the rule's side condition, we have that

$$\Gamma(g(t_1, \ldots, t_k)\theta) \preccurlyeq \Gamma(g(\tau_1, \ldots, \tau_k))$$

  This yields $\Gamma(t_i\theta) \preccurlyeq \tau_i$ for all $i \in \widetilde{k}$. Hence $\theta \Vdash \mathbf{T}(t_i) \preccurlyeq \tau_i$ for all $i \in \widetilde{k}$. Therefore, we have $\theta \Vdash \mathcal{C}_2$.
- $R^{sim}_{\preccurlyeq}$: From the side's condition, we have $t = X$ is a variable, By assumption, we have $\Gamma(X\theta) \preccurlyeq g(\tau_1, \ldots, \tau_k)$. By Lemma 2, there are ground terms $t_1, \ldots, t_k$ such that $X\theta = g(t_1, \ldots, t_k)$ and $\Gamma(t_i) \preccurlyeq \tau_i$ for all $i \in \widetilde{k}$. Since X_is are fresh and pairwise distinct, we define $X_i\theta = t_i$ for all $i \in \widetilde{k}$. Thus, we obtain that

$$\theta \Vdash X \doteq g(X_1, \ldots, X_k)$$
$$\theta \Vdash \mathbf{T}(X_i) \preccurlyeq \tau_i$$

  Therefore $\theta \Vdash \mathcal{C}_2$ as required.

– $R^{\perp}_{\preccurlyeq}$: From the side's condition, we have four cases.
  - If t is atomic and $\neg(\Gamma(t) \preccurlyeq \tau)$ then since $t\theta = t$, we obtain a contradiction from $\Gamma(t\theta) \preccurlyeq \tau$.
  - If $atom(t) \in \mathcal{C}_1$ and τ is composed, then $t\theta$ is an atom. This implies $\neg(\Gamma(t\theta) \preccurlyeq \tau)$ which contradicts $\theta \Vdash \mathbf{T}(t) \preccurlyeq \tau$.
  - If t is composed, $\tau \neq msg$, and τ is atomic, then $\Gamma(t\theta)$ is composed. Thus it is obvious that $\neg(\Gamma(t\theta) \preccurlyeq \tau)$ which yields a contradiction.
  - If t and τ are composed and $top(t) \neq top(\tau)$ then by Lemma 2, we have $\neg(\Gamma(t\theta) \preccurlyeq \tau)$. This contradicts $\theta \Vdash \mathbf{T}(t) \preccurlyeq \tau$.
– $R^{sim}_{\doteq}$: From the rule's side condition, we have $\theta \Vdash t \doteq u$. Hence $t\theta = u\theta$. Since $\sigma = mgu(t, u)$ exists, there is a substitution ξ such that $\theta = \xi \circ \sigma$. Moreover, since θ is ground, we have $X\xi = X\theta$ for all $X \in dom(\xi)$. Thus, ξ is also a model of $\mathcal{C}\sigma$.
– $R^{\perp}_{\doteq}$: Since $\theta \Vdash t \doteq u$, we have $t\theta = u\theta$. This means $mgu(t, u)$ exists which is a contradiction.
– $R^{dup}_{\rhd_f}$: From the rule's side condition, we have

$$eval_f(f(t\theta)) = u\theta$$
$$eval_f(f(t\theta)) = u'\theta$$

Hence $u\theta = u'\theta$. That means $\theta \Vdash u \doteq u'$. Thus, we conclude this case.
– $R^{\perp}_{\rhd_f}$: Since $\theta \Vdash t \rhd_f u$, we have $eval_f(t\theta) = u\theta$. Since $top(t) \neq f$, we have $top(eval_f(t\theta)) = top(t)$. From the side's condition, we know that u is not a variable and $top(u\theta) = top(u)$. We also have $top(t) \neq top(u)$. This implies $top(eval_f(t\theta)) \neq top(u\theta)$ which is a contradiction. Hence, we conclude this case.
– $R^{alg}_{\rhd_f}$: Since θ is a model of $\mathcal{C}_1$, we have $eval_f(f(t\theta)) = u\theta$. Moreover, there exists the first pattern $f(p) = q$ in E^1_f and a ground substitution σ such that

$$dom(\sigma) = vars(p^*),$$
$$\Gamma(t\theta) \preccurlyeq \Gamma(p), \text{ and}$$
$$t\theta = p^*\sigma$$

Since F_f is pattern-disjoint, we have $eval_f(f(t\theta)) = eval_f(q^*\sigma)$. Therefore, we obtain $eval_f(q^*\sigma) = u\theta$. We show that $\sigma \circ \theta$ is a model of some constraint system on the right-hand side. Note that since $\theta \Vdash \mathcal{C}_1$, we have $\theta \Vdash \mathcal{C}$. Moreover, since $dom(\sigma) \cap vars(\mathcal{C}_1) = \emptyset$, we also have $\sigma \circ \theta \Vdash \mathcal{C}$. It remains to show $\sigma \circ \theta$ is a model of the new constraints we have added. Let us consider two cases.
  - If $t\theta$ is composed then $p \notin \mathcal{V}_{pt}$. Since $vars(p^*) \cap vars(\mathcal{C}_1) = \emptyset$, we derive that $p^*(\sigma \circ \theta) = p^*\sigma = t\theta$. Moreover, since θ is ground, we derive that $t(\sigma \circ \theta) = t\theta$. Thus, we have $t(\sigma \circ \theta) = p^*(\sigma \circ \theta)$ and $q^*(\sigma \circ \theta) = q^*\sigma$. It follows that $eval_f(q^*(\sigma \circ \theta)) = eval_f(q^*\sigma) = u\theta$. It follows that $\sigma \circ \theta \Vdash (\mathbf{T}(t) \preccurlyeq \Gamma(p), t \doteq p^*, q^* \rhd_f u)$.
    If $(f(p) = q) \in E_f$ then we are done. Otherwise, suppose that $(f(p) = q) \in E^0_f$. We need to show that $\sigma \circ \theta \Vdash notst(t)$. Since $(f(p) = q) \in E^0_f$, by inspecting the program 1, we have $\theta \Vdash notst(t)$. Moreover, we know that $t\theta = t(\sigma \circ \theta)$. Hence, we obtain $\sigma \circ \theta \Vdash notst(t)$ as desired.

- If $t\theta$ is not composed then we have $eval_f(f(t\theta)) = t\theta = u\theta$. Hence $t(\sigma \circ \theta) = t\theta = u\theta$. Thus, we have $\sigma \circ \theta \Vdash t \doteq u$. It also follows that $t(\sigma \circ \theta)$ is not composed, i.e., $\sigma \circ \theta \Vdash atom(t)$. Therefore, we conclude this case.

Now, we show the direction from right to left. Let $\theta \Vdash \mathcal{C}_2$. We show that $\theta \Vdash \mathcal{C}_1$. Let us perform a case distinction on the reduction rule that has been applied to $\mathcal{C}_1$ (trivial cases are omitted).

- $R^{dup,el}_{\preccurlyeq}$: This case follows from the definition of $lup(\tau, \tau')$ and the transitivity of $\preccurlyeq_0$.
- $R^{split}_{\preccurlyeq}$: Suppose $\theta \Vdash (\mathbf{T}(t_i) \preccurlyeq \tau_i, \mathcal{C})$ for some $i \in \widetilde{k}$. Then we have $\theta \Vdash \mathbf{T}(t_i) \preccurlyeq \tau_i$, which implies

$$\Gamma(g(t_1, \ldots, t_k)\theta) \preccurlyeq \Gamma(g(\tau_1, \ldots, \tau_k))$$

  Hence, we have $\theta \Vdash \mathbf{T}(g(t_1, \ldots, t_k)) \preccurlyeq g(\tau_1, \ldots, \tau_k)$ and conclude this case.
- $R^{sim}_{\preccurlyeq}$: By assumption, we have $X\theta = g(X_1\theta, \ldots, X_k\theta)$. We also have $\Gamma(X_i\theta) \preccurlyeq \tau_i$ for all $i \in \widetilde{k}$. This implies $\Gamma(X\theta) \preccurlyeq g(\tau_1, \ldots, \tau_k)$. Therefore, we have $\theta \Vdash \mathbf{T}(X) \preccurlyeq g(\tau_1, \ldots, \tau_k)$ and conclude this case.
- $R^{elim}_{\preccurlyeq}$: From the side's condition, we have two cases.
  - If $\tau = msg$ then it is obvious that $\Gamma(t\theta) \preccurlyeq \tau$. Therefore, we have $\theta \Vdash \mathbf{T}(t) \preccurlyeq \tau$.
  - If t is atomic and $\Gamma(t) \preccurlyeq \tau$ then we have $t\theta = t$ and $\Gamma(t\theta) \preccurlyeq \tau$. Therefore, we obtain $\theta \Vdash \mathbf{T}(t) \preccurlyeq \tau$ as desired.
- $R^{sim}_{\doteq}$: Since $\theta \Vdash \mathcal{C}\sigma$ and θ is ground, we have $dom(\theta) \cap dom(\sigma) = \emptyset$. Therefore, we derive that $\theta \circ \sigma \Vdash \mathcal{C}$. Moreover, since $\sigma = mgu(t, u)$, we have $t\sigma = u\sigma$. Hence $t(\theta \circ \sigma) = u(\theta \circ \sigma)$. Thus $theta \circ \sigma \Vdash t \doteq u$ which concludes the case.
- $R^{elim}_{\neg,\doteq}$: From the side's condition, we derive that $top(t\theta) \neq top(u\theta)$. Hence $t\theta \neq u\theta$. Therefore, we have $\theta \Vdash \neg(t \doteq u)$ and conclude this case.
- $R^{dup}_{\rhd_f}$: From the rule's side condition, we have $u\theta = u'\theta$ and $eval_f(f(t\theta)) = u\theta$. Hence $eval_f(f(t\theta)) = u'\theta$. Therefore, we have $\theta \Vdash f(t) \rhd_f u'$ as desired.
- $R^{alg}_{\rhd_f}$: Since $\theta \Vdash \mathcal{C}_2$, we have three cases.
  - There is $(f(p) = q) \in E_f$, $p \notin \mathcal{V}_{pt}$ such that $t\theta = p^*\theta$ and $\theta \Vdash (\mathbf{T}(t) \preccurlyeq \Gamma(p), q^* \rhd_f u, \mathcal{C})$. Hence, we have

$$\begin{aligned} &\Gamma(t\theta) \preccurlyeq \Gamma(p) \\ &eval_f(q^*\theta) = u\theta \end{aligned}$$

    Since F_f is pattern-disjoint, we derive that $eval_f(f(t\theta)) = eval_f(q^*\theta)$. This implies $eval_f(f(t\theta)) = u\theta$. Therefore, we have $\theta \Vdash f(t) \rhd_f u$ which concludes this case.
  - There is $(f(p) = q) \in E^0_f$, $p \notin \mathcal{V}_{pt}$ such that $t\theta = p^*\theta$ and $\theta \Vdash (\mathbf{T}(t) \preccurlyeq \Gamma(p), q^* \rhd_f u, \mathcal{C})$. Since $\theta \Vdash notst(t)$ and $p \notin \mathcal{V}_{pt}$, we derive that $eval_f(f(t\theta)) = eval_f(q^*\theta) = u\theta$. Thus, we obtain $\theta \Vdash f(t) \rhd_f u$ as desired.

- $\theta \Vdash (atom(t), t \rhd_f u, \mathcal{C})$. Then we have $t\theta = u\theta$. Since $\theta \Vdash atom(t)$, we have that $t\theta$ is not composed. Therefore, we derive that $eval_f(f(t\theta)) = t\theta = u\theta$. Therefore, we have $\theta \Vdash f(t) \rhd_f u$ as desired.

This completes the proof of the theorem. □

Next, we show that it is possible to construct a model of a solved constraint system.

Lemma 29. *If $\mathcal{C}$ is solved then we can construct one of its models.*

Proof. Our reasoning is based on the following observations.

(i) As rules $R^{split}_{\preccurlyeq}$, $R^{sim}_{\preccurlyeq}$, $R^{elim}_{\preccurlyeq}$, and $R^{single}_{\preccurlyeq}$ are not applicable, for all constraints $\mathbf{T}(t) \preccurlyeq \tau \in \mathcal{C}$, we have that t is a variable and $\tau \neq msg$ atomic and not singleton.

(ii) As rules $R^{split}_{\neg,\preccurlyeq}$, $R^{\perp}_{\neg,\preccurlyeq}$, and $R^{single}_{\neg,\preccurlyeq}$ are not applicable, for all constraints $\neg(\mathbf{T}(t) \preccurlyeq \tau) \in \mathcal{C}$, we have that t is a variable and $\tau \neq msg$ not singleton.

(iii) As rule $R^{sim}_{\doteq}$ is not applicable, there is no constraint $t \doteq u$ in $\mathcal{C}$.

(iv) As rules $R^{split}_{\neg,\doteq}$, $R^{elim}_{\neg,\doteq}$, and $R^{\perp}_{\neg,\doteq}$ are not applicable, for all constraints $\neg(t \doteq u) \in \mathcal{C}$, we have $vars(t) \cap vars(u) = \emptyset$ and t or u is not composed. Moreover, if $u(t)$ is an atom then $t(u)$ is a variable.

(v) As rules R^{elim}_{atom} and $R^{\perp}_{atom}$ are not applicable, for all constraints $atom(t) \in \mathcal{C}$, we have that t is a variable.

(vi) As $R^{split}_{\rhd_f}$ is not applicable, if $t \rhd_f u \in \mathcal{C}$ then t is of the form $f(t')$ for some term t'.

(vii) As $R^{alg}_{\rhd_f}$ is not applicable, if $f(t) \rhd_f u \in \mathcal{C}$ then the following holds.
- t is a variable,
- u is not composed, and
- $cs_{\mathsf{deq}}(\mathcal{C}, t) \cup cs_{\mathsf{type}}(\mathcal{C}, t) = \emptyset$.

(viii) As $R^{dup}_{\rhd_f}$ is not applicable, for a term t, there is at most one constraint $f(t) \rhd_f u \in \mathcal{C}$.

We now show that there is a model θ of $\mathcal{C}$ such that

$$\text{for all } X \in dom(\theta),\ X\theta \text{ is an atom.} \tag{29}$$

We proceed by induction on $vars(\mathcal{C})$.

- If $vars(\mathcal{C}) = \emptyset$ then it is clear that the empty substitution is a model of $\mathcal{C}$ and has the desired property.
- If $vars(\mathcal{C}) \neq \emptyset$ then we consider two cases.
  - If there is a constraint of the form $f(t) \rhd_f u \in \mathcal{C}$ then by (vii), we have $t = X$ is a variable. Moreover, by (viii), there is no other constraint $f(X) \rhd_f u' \in \mathcal{C}$. Let $\mathcal{C}'$ be the set of all the constraints involving X. It is clear that $vars(\mathcal{C} \setminus \mathcal{C}') \subset vars(\mathcal{C})$. Thus, by induction hypothesis, there

is a model θ of $\mathcal{C} \setminus \mathcal{C}'$ such that θ satisfies (29). From (v)-(viii), we derive that

$$\mathcal{C}' = \{f(X) \rhd_f u\} \cup \bigcup_{i=1}^{k}(f(Z_i) \rhd_f X) \cup \bigcup_{i=1}^{n}(atom(Y_i))$$
$$\bigcup_{i=1}^{n} Y_i \subseteq \{X\} \cup \bigcup_{i=0}^{k} Z_i \cup vars(u)$$

where $k, n \geq 0$, u is not composed, and $Y_1, \ldots, Y_n, Z_1, \ldots, Z_k$ are variables. We show that there is a solution σ of $\mathcal{C}'$ and σ satisfies (29). We have $vars(\mathcal{C}') \cap dom(\theta) = \emptyset$. We consider the following cases.

* If u is an atom then we define the substitution σ such that $dom(\sigma) = vars(\mathcal{C}')$ and $V\sigma = \{u\}$. Hence, we have

$$\sigma \Vdash f(X) \rhd_f u$$
$$\sigma \Vdash f(Z_i) \rhd_f X \text{ for all i} \in \widetilde{k}$$
$$\sigma \Vdash atom(Y_i) \text{ for all } i \in \widetilde{n}$$

  Note that there are no other constraints involving any variable in $vars(\mathcal{C}')$. Therefore, we obtain $\sigma \Vdash \mathcal{C}'$. We also have that σ satisfies (29).
* If $u = Z$ is a variable and $Z \in dom(\theta)$ then we define the substitution σ such that $dom(\sigma) = vars(\mathcal{C}')$ and $\sigma(vars(\mathcal{C}')) = \{Z\theta\}$. Similarly as above, we conclude that $\sigma \Vdash \mathcal{C}'$ and σ satisfies (29).
* If $u = Z$ is a variable and $Z \notin dom(\theta)$ then there is no constraint involving Z in $\mathcal{C} \setminus \mathcal{C}'$. Therefore, we define a substitution σ such that $dom(\sigma) = V$ and $\sigma(vars(\mathcal{C}')) = \{a\}$ for an arbitrary atom a. We obtain that $\sigma \Vdash \mathcal{C}'$ and σ satisfies (29).

Now, we define $\theta' = \theta \circ \sigma$. Then θ' is a model of $\mathcal{C}$ and satisfies (29).

- If there is no constraint of the form $f(t) \rhd_f u \in \mathcal{C}$ and there is a constraint of the form $\neg(t \doteq u) \in \mathcal{C}$, then by (iv), we can, without loss of generality, assume that $t = X$ is a variable (otherwise, swapping t and u achieves the desired assumption). Note that in this case, a constraint involving t can only be either a disequality or a type constraint. Therefore, the set $\mathcal{C}'$ of all constraints involving t in $\mathcal{C}$ is given by

$$\mathcal{C}' = cs_{\mathsf{deq}}(\mathcal{C}, t) \cup cs_{\mathsf{type}}(\mathcal{C}, t) \cup cs_{\mathsf{atom}}(\mathcal{C}, t)$$

  Suppose that

$$cs_{\mathsf{deq}}(\mathcal{C}, t) = \{\neg(t \doteq u_1), \ldots, \neg(t \doteq u_k)\} \text{ for } k \geq 0$$

  Then, we have $vars(\mathcal{C} \setminus \mathcal{C}') \subset vars(\mathcal{C})$. By induction hypothesis, there is a model θ of $\mathcal{C}'$. By (i) and (ii), there are infinitely many atoms a such that substituting t by a satisfies $cs_{\mathsf{type}}(\mathcal{C}, t)$. Moreover, since $t \notin vars(u_i)$ for all $i \in \widetilde{k}$ and k is finite, there exists an atom a such that $a \neq u_i\theta$ for all $i \in \widetilde{k}$. Thus, we define the substitution σ such that $dom(\sigma) = vars(\mathcal{C}')$ and

$$Y\sigma = a \text{ for all } Y \in dom(\sigma) \setminus dom(\theta)$$
$$Y\sigma = Y\theta \text{ for all } Y \in dom(\sigma) \cap dom(\theta)$$

It is clear that $\sigma \Vdash \mathcal{C}'$ and satisfies (29). Therefore, we have $\theta \circ \sigma$ is a model of $\mathcal{C}$ and satisfies (29).

- If there are only type constraints involving t or the constraint $atom(t)$ then it is obvious that there exists an atom a such that $\theta' = \theta \cup \{t \mapsto a\}$ satisfies all these constraints. Moreover, we have $\theta' \Vdash \mathcal{C}$ and θ' satisfies (29) as desired.

This completes the proof of the lemma. □

Proposition 9. *Let $t, u \in \mathcal{M}^\sharp$ be terms such that $t, u \in udom(F_f)$ and $msg \notin \Gamma(vars(u))$. Let $\mathcal{C}$ be a constraint system such that*

$$\mathcal{C} = \{\mathbf{T}(X) \preccurlyeq \Gamma(X) \mid X \in vars(u) \cup vars(t)\} \cup \{f(t) \rhd_f eval_f(f(u)), \neg(t \doteq u)\}.$$

Then the following points are equivalent.

(P1) $\mathcal{C} \rightsquigarrow \bot$.
(P2) for all ground substitutions σ that is well-typed such that $f(u\sigma) = f(t\sigma)$, we have $u\sigma = t\sigma$.

Proof. We prove both directions by contradiction.

– $P1 \Rightarrow P2$. Assume there is a ground substitutions σ that is well-typed such that $f(u\sigma) = f(t\sigma)$ and $u\sigma \neq t\sigma$. By Theorem 1, we have $f(u\sigma) = f(u)f(\sigma)$. Since $\Gamma^{-1}(msg) \cap vars(u) = \emptyset$, we know that $f(\sigma) = \sigma$. Therefore, we obtain $f(u\sigma) = f(u)\sigma$. Thus, we have

$$\sigma \Vdash \{f(t) \rhd_f eval_f(f(u)), \neg(t \doteq u)\}.$$

Moreover, since σ is well-typed, we have $\sigma \Vdash \{\mathbf{T}(X) \preccurlyeq \Gamma(X) \mid X \in vars(t) \cap vars(u)\}$. Thus σ is a model of $\mathcal{C}$. This, by Theorem 6, contradicts $P1$.

– $P2 \Rightarrow P1$. Suppose that $P1$ does not hold, i.e., there is a reduction sequence that lead $\mathcal{C}$ to a non-empty solved constraint system $\mathcal{C}'$. By Theorem 6 and Lemma 29, there is a model θ of $\mathcal{C}$. Thus, we have $f(t\theta) = eval_f(f(u))\theta$ and $u\theta \neq t\theta$. By Theorem 1, we have $f(u\theta) = f(u)f(\theta)$. Since $\Gamma^{-1}(msg) \cap vars(u) = \emptyset$, we have $f(\theta|_{vars(u)}) = \theta|_{vars(u)}$. Hence, we obtain that $eval_f(f(u))\theta = f(u)f(\theta) = f(u\theta)$. This yields $f(t\theta) = f(u\theta)$. Moreover, since θ is a model of $\mathcal{C}$, we have

$$\theta \Vdash \{\mathbf{T}(X) \preccurlyeq \Gamma(X) \mid X \in vars(u) \cup vars(t)\}.$$

Thus θ is well-typed. Hence by Theorem 1, we have $f(\theta') = \theta'$. It follows that

$$eval_f(f(t))f(\theta) = eval_f(f(t))f(\theta') = eval_f(f(t))\theta' = eval_f(f(t))\theta = f(u\theta).$$

This implies that $P2$ does not hold and therefore, we conclude the case.

□

F.4 Soundness conditions for $\mathsf{IKE_m}$-to-$\mathsf{IKE_m^1}$ abstraction

Here, we establish the soundness conditions for the abstraction $F_1 = (f_1, E_1)$ in Example 6 with respect to the properties ϕ_s and ϕ_a expressed in Example 4. We assume that $IK_0 = IK_0' = \mathcal{A} \cup \mathcal{C} \cup \mathcal{F}^\bullet \cup \mathsf{sh}(\mathcal{A}_C, \mathcal{A}) \cup \mathsf{sh}(\mathcal{A}, \mathcal{A}_C)$. We need to show all four conditions in Theorem 4 hold. Our argument relies on the following observations.

(O1) $f_1(IK_0) = IK_0 = IK_0'$.
(O2) All terms in $subs(\mathcal{M}_P \cup Sec_\phi \cup EqTerm_\phi)$ are abstracted using only clauses in E_1.
(O3) No term in $subs(\mathcal{M}_P \cup Sec_\phi \cup EqTerm_\phi)$ contains function symbols in D.
(O4) for all terms $t, u \in \mathcal{M}_P$ such that $t \neq u$, we have $f(t) \neq f(u)$.

Condition (1) follows from (O1). Condition (2) follows from Lemma 13. Condition (3) holds by (O2) and (O3). To see that condition (4) holds, we justify three conditions in Definition 10. Condition (i) follows from (O2) and (O3). Condition (iii) holds by (O4). To justify condition (ii), note that we can rewrite the equality on the tuples in ϕ_a as a conjunction of equalities on the tuples' components. Since f_1 is the identity on atoms and variables, it suffices to establish condition (ii) for the two equalities of the form $X = g(a)$ with X is of type msg and a is an atom. We show that condition (ii) holds by Proposition 8.

First, we check conditions (i)-(vi) in Lemma 26. Let $\Pi_f^+ = \Pi(E_f^+)$. Condition (i) holds since no $c \in \mathsf{D}$ occurs in $subs(\Pi_f^+) \setminus \{\Pi_f^+\}$ and $\tau : msg$ for all $\tau \in subs(\Pi_f^+) \setminus \{\Pi_f^+\}$. Conditions (ii) and (iii) hold since no term in $subs(\mathcal{M}_P \cup Sec_\phi \cup EqTerm_\phi)$ contains function symbols in D. Condition (iv) follows from the assumptions. Condition (v) holds by definition of IK_0. Condition (vi) holds by assumptions on R and Ax.

Second, we check conditions (i)-(vi) in Proposition 8. Conditions (i)-(ii) follows the assumptions. Condition (iii) holds because f is an identity on $\mathsf{exp}(g, x^j)$ and $\mathsf{exp}(g, x^j)$ is R,Ax-stable. Condition (iv) holds by definition of E_1. Condition (v) holds since f is homomorphic for exp. Condition (vi) holds since there is no unifier for $\mathsf{exp}(g, x^j)$ and $\mathsf{exp}(\mathsf{exp}(g, V), W)$. Thus, we can invoke Proposition 8 and derive the conclusion.

G Untyped protocol abstractions

Typed protocol abstractions allows simplifying protocols by pulling fields out of an encryptions or removing fields in a hash. However, they do not allow removing a term, e.g., a hash or a variable, completely. In order to abstract a protocol extensively, we introduce *untyped protocol abstractions* which are complementary to typed ones. Informally, an untyped protocol abstraction is a function $f_u : \mathcal{T} \rightharpoonup \mathcal{T} \cup \{\mathsf{nil}\}$ which we extend to events, event sequences, traces, and protocols. Let P be a protocol, we define

(i) $f_u(ev(t)) = ev(f_u(t))$ for events $ev(t) \in Evt(\mathcal{T})$.
(ii) for event sequences, $f(\epsilon) = \epsilon$ and $f_u(e \cdot tl) = f_u(tl)$ if $term(f_u(e)) = \mathsf{nil}$ and $f_u(e \cdot tl) = f_u(e) \cdot f_u(tl)$ otherwise; the lifting to traces is defined analogously.
(iii) $f(P)(X) = P(X)$ for all $X \in dom(P)$ such that $P(X) \neq \epsilon$.

G.1 Redundancy removal abstraction

In this section, we discuss protocol abstractions which allow us to remove redundancies in protocol specifications. For instance, we can remove intruder-derivable terms or repeated occurrences of a term. We call these abstractions *redundancy removal abstractions*.

Definition 13. *A function $rd : \mathcal{M}_P \to \mathcal{M} \cup \{\mathsf{nil}\}$ is a* redundancy removal abstraction *RD_P for a protocol P if, for all $R \in dom(P)$, we have $RD_{rd}(IK_0, P(R))$ where the predicate $RD_{rd}(T, S)$ is inductively defined by the following two rules (where $rd(t)$ is removed from the deducibility conditions if it equals nil).*

$$\frac{}{RD_{rd}(T, \epsilon)} \qquad \frac{RD_{rd}(T \cup \{t\}, r) \quad T, \mathcal{V}_\alpha, rd(t) \vdash_E t \quad T, \mathcal{V}_\alpha, t \vdash_E rd(t)}{RD_{rd}(T, ev(t) \cdot r)}$$

We define $rd(t^{\#i}) = rd(t)^{\#i}$ and lift rd over roles, traces, protocols, and thread pools as expected.

Next, we define our class of redundancy removal abstractions for protocols.

Definition 14 (Redundancy removal abstractions for protocols). *Let $P = (\Gamma_P, S_P)$ be a protocol. The set of redundancy removal abstractions RD_P for P is defined by*

$$RD_P = \{rd : \mathcal{M}_P \to \mathcal{M} \mid \forall R \in dom(S_P).\ RD_{rd}(\Gamma_P, IK_0)S_P(R)\}.$$

For all $rd \in RD_P$, $t \in \mathcal{M}_P$, and $i \in TID$, we define $rd(t^{\#i}) = rd(t)^{\#i}$.

We overload the notation and use $term(tr)$ to denote the set of terms occurring in trace tr. In the following theorem, we show reachability preservation for redundancy removal abstractions.

Lemma 30. *Let P be a protocol and $rd \in RD_P$. Then, for all states (tr, th, σ) reachable in P, we have $IK(rd(tr))\sigma, IK_0 \vdash_E term(tr)\sigma$.*

Proof. We proceed by induction on the number n of transitions leading to a state (tr, th, σ). The theorem trivially holds for base case $(n = 0)$ where tr is the empty trace.

For the inductive case $(n = k + 1)$, we assume that (tr', th', σ) is reachable in k steps and there is a transition $(tr', th', \sigma) \to (tr, th, \sigma)$. Suppose that this transition is performed by thread i. From the transition rules, we know that $tr = tr' \cdot (i, ev(t))$ for some $ev \in \{\mathsf{snd}, \mathsf{rcv}\}$. By the induction hypothesis, we have

$$IK(rd(tr'))\sigma, IK_0 \vdash_E term(tr')\sigma. \tag{30}$$

Since it follows from the induction hypothesis that

$$IK(rd(tr))\sigma, IK_0 \vdash_E term(tr')\sigma$$

Moreover, we have $term(tr) = term(tr') \cup \{t\}$. Thus it is sufficient to show

$$IK(rd(tr))\sigma, IK_0 \vdash_E t\sigma. \tag{31}$$

We do this by case analysis on the rule that justifies transition $k + 1$.

– Rule *SEND*. We have $rd(tr) = rd(tr') \cdot (i, \mathsf{snd}(rd(t)))$ and thus $IK(rd(tr)) = IK(rd(tr')) \cup \{rd(t)\}$ if $rd(t) \neq \mathsf{nil}$ and $rd(tr) = rd(tr')$ otherwise. Hence, we can derive

$$\begin{array}{ll} IK(rd(tr))\sigma, IK_0 \vdash_E IK(rd(tr'))\sigma, rd(t)\sigma, IK_0 & \text{by above} \\ \vdash_E term(tr')\sigma, rd(t)\sigma, IK_0 & \text{by induction hyp. (30)} \end{array}$$

Next, since the terms of all events preceding $\mathsf{snd}(t)$ on $P(R)$ are contained in $term(tr')$ and $rd \in RD_P$, we derive $IK_0, term(tr'), \mathcal{V}_\alpha, rd(t) \vdash_E t$. Instantiating this with σ and observing that $(\mathcal{V}_\alpha)\sigma \subseteq \mathcal{A} \subseteq IK_0$ yields

$$term(tr')\sigma, rd(t)\sigma, IK_0 \vdash_E t\sigma.$$

Combining this with the derivation above yields the desired conclusion (31).

– Rule *RECV*. In this case we can reason as follows.

$$\begin{array}{ll} IK(rd(tr))\sigma, IK_0 \vdash_E term(tr')\sigma, IK_0 & \text{by induction hypothesis (30)} \\ \vdash_E IK(tr')\sigma, IK_0 & \text{since } IK(tr) \subseteq term(tr') \\ \vdash_E t\sigma & \text{by second premise of rule } RECV \end{array}$$

This establishes (31) as required.

This concludes the proof of the lemma. □

Proposition 10. *Let P be a protocol and $rd \in RD_P$. Suppose that $IK_0 \subseteq IK_0'$. Then, for all states (tr, th, σ) reachable in P, $(rd(tr), rd(th), \sigma)$ is a reachable state of $rd(P)$.*

Proof. We proceed by induction on the number n of transitions leading to a state (tr, th, σ). The theorem trivially holds for base case ($n = 0$) where tr is the empty trace. For the inductive case ($n = k + 1$), we assume that (tr', th', σ) is reachable in k steps and there is a transition $(tr', th', \sigma) \to (tr, th, \sigma)$. Suppose that this transition is performed by thread i. From the transition rules, we know that we have $th'(i) = (R, ev(t) \cdot tl)$ where $R \in dom(P)$, $ev \in \{\mathsf{snd}, \mathsf{rcv}\}$, and tl is a suffix of the role $P(R)^{\#i}$. We also have $tr = tr' \cdot (i, ev(t))$ and $th = th'[i \mapsto tl]$.

By the induction hypothesis, we have $(rd(tr'), rd(th'), \sigma)$ is a reachable state of $rd(P)$. If $rd(t) = \mathsf{nil}$ then we are done, since $rd(tr') = rd(tr)$ and $rd(th') = rd(th)$. Otherwise, we have $rd(t) \neq \mathsf{nil}$. In this case, it is sufficient to show that $rd(P)$ has a transition

$$(rd(tr'), rd(th'), \sigma) \to (rd(tr), rd(th), \sigma).$$

We proceed by case distinction on the rule applied to justify step $k + 1$ of P.

- *SEND* rule. The rule's premise requires $th'(i) = (R, \mathsf{snd}(t) \cdot tl)$. Hence, by the definition of $rd(th')$, we have $rd(th')(i) = (R, ev(rd(t)) \cdot rd(tl))$. Moreover, we have $rd(tr) = rd(tr') \cdot (i, \mathsf{snd}(rd(t)))$ and $rd(th) = rd(th')[i \mapsto rd(tl)]$, which by the *SEND* rule justifies the transition above.
- *RECV* rule. This rule's premises require that $th'(i) = (R, \mathsf{rcv}(t) \cdot tl)$ and

  $$IK(tr')\sigma, IK_0 \vdash_E t\sigma.$$

  The rule's conclusion implies that $tr = tr' \cdot (i, \mathsf{rcv}(t))$ and $th = th'[i \mapsto tl]$. In order to apply the *RECV* rule in the state $(rd(tr'), rd(th'), \sigma)$ two premises must be satisfied: first, $rd(th')(i) = (R, \mathsf{rcv}(rd(e)) \cdot rd(tl))$, which holds by the definition of $rd(th')$, and, second,

  $$IK(rd(tr'))\sigma, IK_0' \vdash_E rd(t)\sigma, \tag{32}$$

  which we show now. Since $rd \in RD_P$ and $term(tr')$ contains the terms of all events preceding $\mathsf{rcv}(t)$ on $P(R)$, we have $IK_0, term(tr'), \mathcal{V}_\alpha, t \vdash_E rd(t)$. Noting that $term(tr) = term(tr') \cup \{t\}$ and $(\mathcal{V}_\alpha)\sigma \subseteq \mathcal{A} \subseteq IK_0$ we derive

  $$term(tr)\sigma, IK_0 \vdash_E rd(t)\sigma.$$

  Moreover, from Lemma 30, we have $IK(rd(tr))\sigma, IK_0 \vdash_E term(tr)\sigma$. Combining these facts with the observation that $IK(rd(tr)) = IK(rd(tr'))$ and the assumption that $IK_0 \subseteq IK_0'$, we obtain (32) as required.

This completes the proof of the theorem. □

Next, we extend rd to formulas $\phi \in \mathcal{L}_P$ as follows:

$$\begin{aligned}
rd((i = i')) &= (i = i') & rd(secret(m)) &= secret(m)\\
rd((m = m')) &= (m = m') & rd(\neg A) &= \neg rd(A)\\
rd(role(i, R)) &= role(i, R) & rd(\phi_1 \wedge \phi_2) &= rd(\phi_1) \wedge rd(\phi_2)\\
rd(honest(i, R)) &= honest(i, R) & rd(\phi_1 \vee \phi_2) &= rd(\phi_1) \vee rd(\phi_2)\\
rd(steps(i, e)) &= steps(i, rd(e)) & rd(\forall i.\, \phi') &= \forall i.\, rd(\phi')\\
rd((i, e) \prec (j, e')) &= (i, rd(e)) \prec (j, rd(e')) & rd(\exists i.\, \phi') &= \exists i.\, rd(\phi')
\end{aligned}$$

Let $rd \in RD_P$. In the following, we define the notion of (P, rd)-safe formulas for which the attack preservation holds.

Definition 15 ((P, rd)-safe formulas). *Let P be a protocol and rd a redundancy abstraction for P. A formula ϕ is* (P, rd)-safe *if*

1. *$rd(t) = rd(u)$ implies $t = u$, for all $e(t) \in Evt^+_\phi$ and $e(u) \in Evt(\mathcal{M}_P)$,*
2. *for all $ev(t) \in Evt_\phi$, we have $rd(t) \neq \mathsf{nil}$.*

Theorem 7 (Soundness for redundancy removal abstractions). *Let P be a protocol, $\phi \in \mathcal{L}_P$, and $rd \in RD_P$ be a (P, rd)-safe formula. Suppose that $IK_0 \subseteq IK'_0$. Then, for all states (tr, th, σ) reachable in P, we have*

1. *$(rd(tr), rd(th), \sigma)$ is reachable in $rd(P)$ and*
2. *$(tr, th, \sigma) \not\models \phi$ implies $(rd(tr), rd(th), \sigma) \not\models rd(\phi)$.*

Proof. By Proposition 10, we have that $(rd(tr), rd(th), \sigma)$ is reachable in $rd(P)$. It remains to show that

$$\forall \vartheta.\ (tr, th, \sigma, \vartheta) \not\models \phi \Rightarrow (rd(tr), rd(th), \sigma) \not\models rd(\phi).$$

We proceed by induction on the structure of ϕ and consider the following cases.

– $\phi \equiv m = m'$ or $\phi \equiv \neg(m = m')$.

$$\begin{aligned} & (tr, th, \sigma, \vartheta) \vDash m =_E m' \\ \Leftrightarrow\ & m\sigma =_E m'\sigma \\ \Leftrightarrow\ & (rd(tr), rd(th), \sigma, \vartheta) \vDash rd((m = m')) \end{aligned}$$

– $\phi = secret(m)$.

$$\begin{aligned} & (tr, th, \sigma, \vartheta) \not\models secret(m) \\ \Leftrightarrow\ & IK(tr)\sigma, IK_0 \vdash_E m\sigma \\ \Rightarrow\ & IK(rd(tr))\sigma, IK_0 \vdash_E m\sigma && \text{by Lemma 30} \\ \Leftrightarrow\ & (rd(tr), rd(th), \sigma, \vartheta) \not\models rd(secret(m)) \end{aligned}$$

The remaining cases are routine. This completes the proof of the theorem. □

G.2 Variable removal abstractions

Our typed abstractions do not allow us to remove message variables. This is because our framework heavily relies on the substitution property, which fails to hold for the removal of variables. In this section, we establish a soundness result for *variable removal abstractions* separately for a subclass of protocols, namely *well-formed protocols*.

Definition 16. *A protocol P is* well-formed *if all non-agent variables first occur in receive events, i.e., for all events e in a role $P(R)$ and all variables $X \in vars(term(e))$ such that $\Gamma(X) \neq \alpha$, there is an event $\mathsf{rcv}(t)$ in $P(R)$ such that $\mathsf{rcv}(t)$ equals or precedes e in $P(R)$ and $X \in vars(t)$.*

Below we identify some conditions under which the set of messages the intruder can derive (and hence any potential attack) is preserved by variable removal. Before stating these assumptions, we introduce some auxiliary definitions.

First, we extend the notion of clearness to a set of terms T as expected, i.e., T is *clear* in a term t if every term in T is. We also say that T is clear in a set of terms T' if every $t \in T$ is clear in every $t' \in T'$. In the following lemma, we abuse the notation and use $vars(tr)$ to denote the set of variables occurring in tr.

Lemma 31. *Let P be a well-formed protocol and (tr, th, σ) be a reachable state of P. Let $V \subseteq \mathcal{V}$ be a set of variables such that V is clear in $\mathcal{M}_P$. Then we have that $rem_V(IK(tr))\sigma, IK_0 \vdash_E (IK(tr) \cup (vars(tr) \cap V^\flat))\sigma$.*

Proof. We proceed by induction on tr. For the base case, $tr = \epsilon$, the lemma holds trivially. For the inductive step, suppose (tr', th', σ) is reachable in P and there is a transition $(tr', th', \sigma) \to (tr, th, \sigma)$ such that $tr = tr' \cdot (i, ev(t))$ for some $i \in TID$ and some term t. By induction hypothesis, we have

$$rem_V(IK(tr'))\sigma, IK_0 \vdash_E (IK(tr') \cup (vars(tr') \cap V^\flat))\sigma.$$

and we have to show $rem_V(IK(tr))\sigma, IK_0 \vdash_E (IK(tr) \cup (vars(tr) \cap V^\flat))\sigma$. We reason by a case distinction on the rule r that has been applied in the last step.

- If $r = RECV$ then we have that $IK(tr') = IK(tr)$. Thus by induction hypothesis, we have $rem_V(IK(tr))\sigma, IK_0 \vdash_E IK(tr)\sigma$. Therefore, it remains to show that $rem_V(IK(tr))\sigma, IK_0 \vdash_E (vars(tr) \cap V^\flat)\sigma$.
  Note that $tr = tr' \cdot (i, \mathsf{rcv}(t))$. If $vars(t) \cap V^\flat \subseteq vars(tr')$ then $vars(tr) \cap V^\flat = vars(tr') \cap V^\flat$ and the conclusion follows directly from the induction hypothesis. Otherwise, let $X^i \in (vars(t) \cap V^\flat) \setminus vars(tr')$. Given the induction hypothesis, it is sufficient to establish $rem_V(IK(tr))\sigma, IK_0 \vdash_E X^i\sigma$.
  By the premises of the $RECV$ rule, we know that $IK(tr')\sigma, IK_0 \vdash_E t\sigma$. Since V is clear in t, we also have

$$IK(tr')\sigma, IK_0 \vdash_E X^i\sigma \tag{33}$$

  Since V is clear in $IK(tr')$, we have

$$IK_0, (V^\flat \cap vars(IK(tr'))), rem_V(IK(tr')) \vdash_E IK(tr').$$

By instantiating this with σ and using the fact that $vars(IK(tr')) \subseteq vars(tr')$, we obtain

$$IK_0, (V^\flat \cap vars(tr'))\sigma, rem_V(IK(tr'))\sigma \vdash_E IK(tr')\sigma$$

Together with the induction hypothesis and $IK(tr) = IK(tr')$, we derive

$$rem_V(IK(tr))\sigma, IK_0 \vdash_E IK(tr')\sigma.$$

Combining this with (33), we obtain $rem_V(IK(tr))\sigma, IK_0 \vdash_E X^i\sigma$ as required.

– If $r = SEND$ then we have $tr = tr' \cdot (i, \mathsf{snd}(t))$. Thus, we have that $IK(tr) = IK(tr') \cup \{t\}$. By the well-formedness of P, we have $vars(tr) = vars(tr')$. Hence, it follows from the induction hypothesis that

$$rem_V(IK(tr))\sigma, IK_0 \vdash_E (vars(tr) \cap V^\flat)\sigma. \tag{34}$$

We are left to show that $rem_V(IK(tr))\sigma, IK_0 \vdash_E IK(tr)\sigma$. Since V is clear in t, we obtain

$$IK_0, (V^\flat \cap vars(t))\sigma, rem_V(t)\sigma \vdash_E t\sigma. \tag{35}$$

Since $vars(t) \subseteq vars(tr)$. By (34), we have

$$rem_V(IK(tr))\sigma, IK_0 \vdash_E (vars(t) \cap V^\flat)\sigma. \tag{36}$$

Together with (36) and (35), we derive that

$$rem_V(IK(tr))\sigma, IK_0, rem_V(t)\sigma \vdash_E t\sigma.$$

Since $t \in IK(tr)$, we have that $rem_V(t)\sigma \in rem_V(IK(tr))\sigma$. Hence, we obtain that

$$rem_V(IK(tr))\sigma, IK_0 \vdash_E t\sigma$$

By induction hypothesis, we have $rem_V(IK(tr')\sigma), IK_0 \vdash_E IK(tr')\sigma$. Hence, we derive that $rem_V(IK(tr))\sigma, IK_0 \vdash_E IK(tr)\sigma$ as required.

This completes the proof of the lemma. □

Proposition 11. *Let P be a well-formed protocol. Suppose V is a set of variables and u a term such that V is clear in $\mathcal{M}_P$. Suppose (tr, th, σ) is a reachable state of P. Then $IK(tr)\sigma, IK_0 \vdash_E u\sigma$ implies $IK(rem_V(tr))\sigma, IK_0 \vdash_E rem_V(u)\sigma$.*

Proof. We derive

$$\begin{aligned} IK(rem_V(tr))\sigma, IK_0 &\vdash_E rem_V(IK(tr'))\sigma, IK_0 \\ &\vdash_E IK(tr)\sigma, IK_0 && \text{by Lemma 31} \\ &\vdash_E u\sigma, IK_0 && \text{by assumption} \\ &\vdash_E rem_V(u)\sigma \end{aligned}$$

The last step follows the assumption that V is clear in $\mathcal{M}_P$. □

The following theorem states the reachability preservation result for variable removal abstractions.

Theorem 8. *Let P be a well-formed protocol. Suppose $IK_0 \subseteq IK_0'$ and $V \subseteq \mathcal{V}$ is a set of variables such that*

1. *V is clear in all terms t occurring in receive events in P,*
2. *V is clear in $\mathcal{M}_P$.*

Let (tr, th, σ) be reachable in P. Then we have that $(rem_V(tr), rem_T(th), \sigma)$ is reachable in $rem_T(P)$.

Proof. We prove the reachability of the state $(rem_V(tr), rem_V(th), \sigma)$ by induction on the number n of transitions leading to a state (tr, th, σ). The theorem holds trivially for the empty trace ($n = 0$). For the inductive case ($n = k + 1$), assume that (tr', th', σ) is reachable in k steps and there is a transition $(tr', th', \sigma) \to (tr, th, \sigma)$. By induction hypothesis, the state $(rem_V(tr'), rem_V(th'), \sigma)$ is reachable in $rem_V(P)$.

We distinguish two cases according to the rule r used to justify the step $k+1$ in P. We first treat the case of the receive rule ($r = RECV$). The rule's premises require that there are $i \in TID$, $R \in dom(P)$, and a suffix tl of the role $P(R)^{\#i}$ such that

$$th'(i) = (R, \mathsf{rcv}(t) \cdot tl) \quad \text{and} \quad IK(tr')\sigma, IK_0 \vdash t\sigma \tag{37}$$

The rule's conclusion implies that $tr = tr' \cdot (i, \mathsf{rcv}(t))$ and $th = th'[i \mapsto tl]$. We consider two cases.

- If $rem_V(t) = \mathsf{nil}$ then we have that

  $$\begin{aligned} rem_V(tr) &= rem_V(tr'), \\ rem_V(th) &= rem_V(th'). \end{aligned}$$

  Hence, we conclude that $(rem_V(tr), rem_V(th), \sigma)$ is reachable in $rem_V(P)$ by the induction hypothesis.
- If $rem_V(t) \neq \mathsf{nil}$ then we show that $rem_V(P)$ has a transition

  $$(rem_V(tr'), rem_V(th'), \sigma) \to (rem_V(tr), rem_V(th), \sigma).$$

  In order to apply the $RECV$ rule in the state $(rem_V(tr'), rem_V(th'), \sigma)$ the following two premises must be satisfied:

  $$\begin{aligned} &rem_V(th')(i) = (R, \mathsf{rcv}(rem_V(t)) \cdot rem_V(tl)), \text{ and} \\ &IK(rem_V(tr')))\sigma, IK_0' \vdash_E rem_V(t)\sigma. \end{aligned}$$

  The first premise holds by application of rem_T to th'. The second one follows from Proposition 11 and the assumption that $IK_0 \subseteq IK_0'$. The successor state in the conclusion of the $RECV$ rule is

  $$(rem_V(tr') \cdot (i, \mathsf{rcv}(rem_V(t))),\ rem_V(th')[i \mapsto (R, rem_V(tl))], \sigma),$$

  which is identical to the state $(rem_V(tr), rem_V(th), \sigma)$, whose reachability in $rem_V(P)$ we have hereby established.

The case of the send rule ($r = SEND$) is similar but simpler, since the deducibility condition falls away. This completes the proof of the theorem. □

Definition 17 ((P, rem_T)-safe formulas). *Let P be a protocol and let $V \subseteq \mathcal{V}$ be a set of variables. A formula $\phi \in \mathcal{L}_P$ is (P, rem_T)-safe iff the following holds.*

1. *for all $(m, m') \in Eq_\phi$, we have $vars(\{m, m'\}) \cap V^\sharp = \emptyset$,*
2. *for all terms $t \in Sec_\phi$, we have $vars(t) \cap V^\sharp = \emptyset$,*
3. *for all events $e(t) \in Evt_\phi$, we have $rem_V(t) \neq \mathsf{nil}$, and*
4. *$rem_V(m) = rem_V(m')$ implies $m = m'$ for all $s(m) \in Evt^+_\phi$ and $s(m') \in Evt(\mathcal{M}_P)$.*

We now state our soundness theorem as follows.

Theorem 9 (Soundness for variable removal abstractions). *Let P be a well-formed protocol, $V \subseteq \mathcal{V}$ is a set of variables that is clear in $\mathcal{M}_P$. Let $\phi \in \mathcal{L}_P$ be a (P, rem_T)-safe formula and (tr, th, σ) be reachable in P. Suppose that $IK_0 \subseteq IK_0'$. Then we have that $(rem_V(tr), rem_V(th), \sigma)$ is a reachable state in $rem_V(P)$. Moreover, if $(tr, th, \sigma) \not\models \phi$ then $(rem_V(tr), rem_V(th), \sigma) \not\models \phi$.*

Proof. By Theorem 8, we have that $(rem_V(tr), rem_V(th), \sigma)$ is a reachable state in $rem_V(P)$. It remains to show that

$$\forall \vartheta.\ (tr, th, \sigma, \vartheta) \not\models \phi \Rightarrow (rem_T(tr), rem_T(th), \sigma) \not\models rem_T(\phi).$$

We proceed by induction on the structure of ϕ and consider the following non-trivial cases.

– $\phi \equiv m = m'$ or $\phi \equiv \neg(m = m')$.

$$\begin{aligned}
&(tr, th, \sigma, \vartheta) \models m = m' \\
\Leftrightarrow\ & m\sigma =_E m'\sigma \\
\Leftrightarrow\ & rem_T(m)\sigma =_E rem_T(m')\sigma \qquad \text{(since } \phi \text{ is } (P, V, rem_T)\text{-safe)} \\
\Leftrightarrow\ & (rem_T(tr), rem_T(th), \sigma, \vartheta) \models rem_T(m) = rem_T(m')
\end{aligned}$$

– $\phi = secret(m)$.

$$\begin{aligned}
&(tr, th, \sigma, \vartheta) \not\models secret(m) \\
\Leftrightarrow\ & IK(tr)\sigma, IK_0 \vdash_E m\sigma \\
\Rightarrow\ & rem_T(IK(tr))\sigma, IK_0 \vdash_E m\sigma && \text{by Proposition 11} \\
\Leftrightarrow\ & rem_T(IK(tr))\sigma, IK_0 \vdash_E rem_T(m)\sigma && \text{since } rem_T(m) = m \\
\Rightarrow\ & IK(rem_T(tr))\sigma, IK_0' \vdash_E rem_T(m)\sigma \\
\Leftrightarrow\ & (rem_T(tr), rem_T(th), \sigma, \vartheta) \not\models secret(rem_T(m))
\end{aligned}$$

This completes the proof of the theorem. □

G.3 Atom removal abstraction

Atom removal abstractions allow us to eliminate atoms in clear. This cannot be done by typed or variable removal abstractions. Before we prove deducibility preservation for our atom removal abstractions, we introduce some auxiliary definitions. For any term t, we use $\mathit{atoms}(t)$ and $\mathit{fresh}(t)$ to denote the set of atoms and fresh values occurring in t, respectively. Recall that the attacker has a countably infinite choice of nonces $n_i^\bullet \in \mathcal{F}^\bullet$ for each type β_n. Given a set of atoms $At \subseteq \mathcal{A} \cup \mathcal{C} \cup \mathcal{F}$ and a set of terms T, we define $\mathit{imap}(At, T)$ as the set of injective maps $\rho : \mathit{tID}(At, T) \to \mathbb{N}$ where $\mathit{tID}(At, T) = \{i \in \mathit{TID} \mid f^i \in \mathit{fresh}(At^\sharp) \cap \mathit{split}(T)\}$. We extend ρ to atoms and variables as follows.

- $\rho(n^i) = n^\bullet_{\rho(i)}$ for all $n^i \in \mathit{fresh}(At^\sharp) \cap \mathit{split}(T)$,
- $\rho(u) = u$ for all remaining atomic messages $u \in \mathcal{A} \cup \mathcal{C} \cup \mathcal{V}^\flat \cup \mathcal{F}^\flat \cup \mathcal{F}^\bullet$.

We extend ρ homomorphically to all terms. For the sake of uniformity, we treat ρ as a substitution. For terms t and u, we use $[t/u]$ to denote the replacement of u by t.

Lemma 32. *Let t, u be terms, a be an atom, and σ be a substitution such that*

(i) $\mathit{vars}(t) \cap \mathit{dom}(\sigma) = \emptyset$,
(ii) $a \notin \mathit{vars}(u)$.

Then $(u\sigma)[t/a] = (u[t/a])(\sigma[t/a])$.

Proof. We prove this lemma by induction on u.

- If u is an atom then $(u\sigma)[t/a] = u[t/a]$. Moreover, by (i), it follows that $(u[t/a])(\sigma[t/a]) = u[t/a]$. Thus the lemma holds for this case.
- If u is a variable then by (ii), we have $u[t/a] = u$. Therefore, we have $(u\sigma)[t/a] = u(\sigma[t/a]) = (u[t/a])(\sigma[t/a])$.
- If $u = g(u_1, \ldots, u_n)$ for $g \in \Sigma^n$, $n \geq 1$, then we have

$$\begin{aligned}
(u\sigma)[t/a] &= g((u_1\sigma)[t/a], \ldots, (u_n\sigma)[t/a]) && \text{since } a \text{ is an atom} \\
&= g((u_1[t/a])(\sigma[t/a]), \ldots, (u_n[t/a])(\sigma[t/a])) && \text{by IH} \\
&= (g(u_1, \ldots, u_n)[t/a])(\sigma[t/a]) && \text{since } a \text{ is an atom} \\
&= (u[t/a])(\sigma[t/a])
\end{aligned}$$

This completes the proof of the lemma. □

Next, we prove a deducibility lemma for atom removal abstractions.

Lemma 33. *Let T be set of normal-form terms, t a normal-form term, At a set of atoms such that*

(i) $At^\sharp$ *is clear in* $T \cup \{t\}$, *and*
(ii) $At^\sharp \cap \mathit{fresh}(IK_0) = \emptyset$.

Let σ be a substitution such that $T\sigma, IK_0 \vdash_E t\sigma$. Then, for all $\rho \in \mathit{imap}(At, T \cup \{t\})$, we have

$$\mathit{rem}_{At}(T)(\rho \circ \sigma), IK_0 \vdash_E \mathit{rem}_{At}(t)(\rho \circ \sigma).$$

Proof. We start by observing that under assumption (i) we have

$$rem_{At}(u), \mathcal{F}^\bullet \vdash_E u\rho \quad \text{and} \quad u\rho \vdash_E rem_{At}(u) \tag{38}$$

for all terms $u \in T \cup \{t\}$ and $\rho \in imap(At, T \cup \{t\})$.

Suppose $T\sigma, IK_0 \vdash_E t\sigma$. We show that $rem_{At}(T)\theta, IK_0 \vdash_E rem_{At}(t)\theta$ where $\theta = \rho \circ \sigma$. Together with $\mathcal{F}^\bullet\theta = \mathcal{F}^\bullet$ and $\mathcal{F}^\bullet \subseteq IK_0$, we can use Lemma 23 to derive the following from (38):

$$rem_{At}(T)\theta, IK_0 \vdash_E (T\rho)\theta \quad \text{and} \quad (t\rho)\theta \vdash_E rem_{At}(t)\theta \tag{39}$$

Since $IK_0\rho = IK_0$ by assumption (ii), we can also apply Lemma 23 to the assumption $T\sigma, IK_0 \vdash_E t\sigma$ and deduce $(T\sigma)\rho, IK_0 \vdash_E (t\sigma)\rho$. Next, we use Lemma 32 to deduce $(T\rho)\theta, IK_0 \vdash_E (t\rho)\theta$. Combining this with (39) above yields the desired result $rem_{At}(T)\theta, IK_0 \vdash_E rem_{At}(t)\theta$. □

Theorem 10 (Reachability preservation for atom removal abstractions). *Let P be a protocol and $At \subseteq atoms(\mathcal{M}_P)$ a set of atoms such that*

(i) At is clear in $\mathcal{M}_P$, and
(ii) $At^\sharp \cap fresh(IK_0) = \emptyset$.

Let (tr, th, σ) be a reachable state of P and $\rho \in imap(At, term(tr))$. Then the state $(rem_{At}(tr), rem_{At}(th), \rho \circ \sigma)$ is a reachable state of $rem_{At}(P)$.

Proof. Let $\theta = \rho \circ \sigma$. It is clear that θ is well-typed. We now prove the first conclusion by induction on the number n of transitions leading to a state (tr, th, σ). For the empty trace ($n = 0$), the theorem holds trivially. For the inductive case ($n = k+1$), assume that (tr', th', σ) is reachable in k steps and there is a transition $(tr', th', \sigma) \to (tr, th, \sigma)$. By induction hypothesis, $(rem_{At}(tr'), rem_{At}(th'), \theta)$ is reachable in $rem_{At}(P)$.

There are two cases according to the rule r that has been applied in the step $k+1$ in P. The case of the send rule is easy. We consider the case of receive rule. The rule's premises require that there are $i \in TID$, $R \in dom(P)$, and a suffix tl of the role $P(R)^{\#i}$ such that

$$th'(i) = (R, \mathsf{rcv}(t) \cdot tl) \quad \text{and} \quad IK(tr')\sigma, IK_0 \vdash t\sigma \tag{40}$$

The rule's conclusion implies that $tr = tr' \cdot (i, \mathsf{rcv}(t))$ and $th = th'[i \mapsto tl]$. In order to apply the *RECV* rule in the state $(rem_{At}(tr'), rem_{At}(th'), \theta)$, we must show the following two premises

$$rem_{At}(th')(i) = (R, \mathsf{rcv}(rem_{At}(t)) \cdot rem_{At}(tl)), \text{ and}$$
$$IK(rem_{At}(tr')))\theta, IK_0' \vdash_E rem_{At}(t)\theta.$$

Clearly, the first premise is satisfied by application of rem_{At} to th'. We now show the second one. Using Lemma 33, we deduce from (40) that

$$rem_{At}(IK(tr'))\theta, IK_0 \vdash_E rem_{At}(t)\theta$$

Since $rem_{At}(IK(tr')) = IK(rem_{At}(tr'))$, we obtain

$$IK(rem_{At}(tr'))\theta, IK_0 \vdash_E rem_{At}(t)\theta.$$

Moreover, the successor state in the conclusion of the $RECV$ rule is

$$(rem_{At}(tr') \cdot (i, \mathsf{rcv}(rem_{At}(t))),\ rem_{At}(th')[i \mapsto (R, rem_{At}(tl))], \sigma),$$

which is identical to the state $(rem_{At}(tr), rem_{At}(th), \sigma)$. This concludes the proof of the theorem. □

Definition 18 ((P, rem_{At})-safe formulas). *Let P be a protocol. We assume a set of atoms $At \subseteq atoms(\mathcal{M}_P)$. A formula $\phi \in \mathcal{L}_P$ is* (P, rem_{At})-safe *if the following conditions holds:*

(i) $At^\sharp \cap fresh(Sec_\phi \cup EqTerm_\phi) = \emptyset$,
(ii) for all $e(t) \in Evt_\phi$, *we have* $rem_{At}(t) \neq \mathsf{nil}$, *and*
(iii) $rem_{At}(m) = rem_{At}(m')$ *implies* $m = m'$ *for all* $s(m) \in Evt^+_\phi$ *and* $s(m') \in Evt(\mathcal{M}_P)$.

We now show soundness for atom removal abstractions.

Theorem 11 (Soundness for atom removal abstractions). *Let P be a protocol, $At \subseteq atoms(\mathcal{M}_P)$ and $\phi \in \mathcal{L}_P$ a formula such that*

(i) At is clear in $\mathcal{M}_P$,
(ii) $At^\sharp \cap fresh(IK_0) = \emptyset$, *and*
(iii) ϕ *is* (P, rem_{At})*-safe*

Then, for all reachable states (tr, th, σ) of P, there is a ground substitution θ such that

1. *the state* $(rem_{At}(tr), rem_{At}(th), \theta)$ *is reachable in* $rem_{At}(P)$, *and*
2. $(tr, th, \sigma) \not\models \phi$ *implies* $(rem_{At}(tr), rem_{At}(th), \theta) \not\models rem_{At}(\phi)$.

Proof. Let $\rho \in imap(At, term(tr))$ such that

$$\rho \text{ is injective on } fresh(ran(\sigma)). \tag{41}$$

Let $\theta = \rho \circ \sigma$. By Theorem 10, we have that $(rem_{At}(tr), rem_{At}(th), \theta)$ is reachable in $rem_{At}(P)$ (and thus θ is well-typed). Hence, it remains to show that

$$\forall \vartheta.\ (tr, th, \sigma, \vartheta) \not\models \phi \Rightarrow (rem_{At}(tr), rem_{At}(th), \theta) \not\models rem_{At}(\phi).$$

We proceed by induction on the structure of ϕ and consider the following non-trivial cases.

– $\phi \equiv m = m'$ or $\phi \equiv \neg(m = m')$.

$$\begin{aligned}
&(tr, th, \sigma, \vartheta) \models m = m' \\
\Leftrightarrow\ & m\sigma =_E m'\sigma \\
\Leftrightarrow\ & (m\sigma)\rho =_E (m'\sigma)\rho && \text{(by (41) and (iii))} \\
\Leftrightarrow\ & (m\rho)\theta =_E (m'\rho)\theta && \text{(by Lemma 32)} \\
\Leftrightarrow\ & rem_{At}(m)\theta =_E rem_{At}(m')\theta && \text{(by Def 18(ii))} \\
\Leftrightarrow\ & (rem_{At}(tr), rem_{At}(th), \theta, \vartheta) \models rem_{At}(m) = rem_{At}(m')
\end{aligned}$$

– $\phi = secret(m)$.

$$
\begin{aligned}
& (tr, th, \sigma, \vartheta) \not\models secret(m) \\
\Leftrightarrow\ & IK(tr)\sigma, IK_0 \vdash_E m\sigma \\
\Rightarrow\ & rem_{At}(IK(tr))\theta, IK_0 \vdash_E rem_{At}(m)\theta \quad \text{(by Lemma 33, Def 18(i))} \\
\Rightarrow\ & IK(rem_{At}(tr))\theta, IK_0 \vdash_E rem_{At}(m)\theta \\
\Leftrightarrow\ & (rem_{At}(tr), rem_{At}(th), \theta, \vartheta) \not\models secret(rem_{At}(m))
\end{aligned}
$$

This completes the proof of the theorem. □

H Experimental results

H.1 Scyther tool

The Scyther tool is based on symbolic backwards search and supports verification of both a bounded and an unbounded number of threads. We have demonstrated our abstraction method on a variety of protocols, mostly from the IKE and ISO/IEC 9798 families. Our results with the Scyther tool (version 1.1.2) are summarized in Table 2. Our experiments show substantial performance gains. The abstractions enable Scyther to verify seven protocols (four from the ISO/IEC 9798, two from the IKE families, and the PANA-AKA protocol) for an unbounded number of threads. Remarkably, five of them were verified (at the most abstract levels) within 0.4 seconds whereas it fails (TO) or runs out of memory (ME) on the original protocols.

For the IKE protocols, we approximate the Diffie-Hellman equations in Scyther using oracle roles. This complicates the verification task and, as a consequence, the average performance gain appears to be smaller than with the ISO/IEC protocols or the PANA-AKA protocol. In particular, the unbounded verification of (abstractions of) the first six IKE protocols in Table 2 still results in a timeout. However, we are able to either significantly improve the bounds on the number of threads that can be covered for these protocols or, for IKEv2-eap and IKEv2-eap2, enable protocol verification for a bounded number of threads where it timed out before even for three threads.

Apart from the dramatic speedups we achieve in most cases, we also observe that for many protocols the verification time increases much slower than their originals. For the last seven protocols in the table, this time is almost constant whereas it can grow rapidly, e.g., for ISO/IEC 9798-3-6-1 and PANA-AKA. Moreover, our abstractions greatly reduce memory consumption. In particular, Scyther runs out of memory for ISO/IEC 9798-3-6-1 and ISO/IEC 9798-3-7-1 after 5 and 6 threads, respectively. However, abstraction enables it to run up to an unbounded number of threads.

Scyther find attacks on the most abstract models much faster it does on the originals. Concretely, it falsifies the most abstract model of the IKEv1-sig-m and IKEv1-sig-m-perlman protocols for an unbounded number of threads within 0.06 seconds, while finding the same attacks on the original protocols takes 0.44 and 109 seconds, respectively.

protocol/prop./#threads	No	S	A	W	N	3	4	5	6	7	8	∞
IKEv1-pk2-a2	1	✓			✓	44.01	302.86	1843.80	10999.80	TO	TO	TO
						6.01	26.36	151.88	1014.63	6838.97	TO	TO
IKEv1-pk2-a	1	✓			✓	1103.63	27808.72	TO	TO	TO	TO	TO
						133.65	3356.59	TO	TO	TO	TO	TO
IKEv1-pk-a2	1	✓			✓	10.95	61.47	125.25	237.76	409.35	744.75	TO
						0.84	1.79	2.43	3.63	6.01	9.61	TO
IKEv1-pk-a22	1	✓			✓	18.48	82.93	249.55	554.09	1006.04	1734.85	TO
						0.83	1.26	2.08	3.47	5.96	10.28	TO
IKEv2-eap	5	✓			✓	TO	TO	TO	TO	TO	TO	TO
						78.35	798.44	4212.71	20911.20	TO	TO	TO
IKEv2-eap2	5	✓			✓	TO	TO	TO	TO	TO	TO	TO
						70.18	690.26	4169.87	20071.45	TO	TO	TO
IKEv2-mac	5	✓			✓	1.85	4.91	6.72	8.07	8.42	8.49	8.70
						0.62	1.77	1.83	1.73	1.73	1.80	1.74
IKEv2-mac2	5	✓			✓	2.16	4.09	6.43	9.41	8.16	8.44	8.69
						0.81	1.60	1.73	1.75	1.73	1.74	1.73
IKEv2-mactosig	4	✓			✓	11.65	141.37	1075.46	7440.81	TO	TO	TO
						2.89	12.38	24.54	38.68	53.36	65.07	77.68
IKEv2-mactosig2	4	✓			✓	11.71	133.20	1064.30	7229.13	TO	TO	TO
						2.85	11.81	24.14	38.22	53.25	64.51	77.03
IKEv2-sigtomac	5	✓			✓	6.15	33.19	65.05	115.34	204.93	206.45	237.34
						3.59	12.72	28.44	44.44	55.11	66.97	67.15
IKEv1-pk-m	2				×	48.62	269.92	507.40	869.23	16254.80	TO	TO
						0.04	0.05	0.05	0.05	0.05	0.05	TO
IKEv1-pk-m2	2				✓/×	18.26	274.87	4438.72	TO	TO	TO	TO
						1.48	7.79	32.75	110.32	339.93	963.08	TO
IKEv1-sig-m	2				×	0.34	0.45	0.45	0.45	0.45	0.46	0.44
						0.05	0.05	0.05	0.06	0.05	0.05	0.06
IKEv1-sig-m-perlman	2				×	2.86	13.99	40.78	67.83	72.08	72.15	109.03
						0.05	0.05	0.05	0.05	0.05	0.05	0.05
ISO/IEC 9798-2-5	1	✓				0.78	8.96	73.87	564.67	4214.22	TO	TO
						0.07	0.11	0.12	0.11	0.11	0.11	0.11
ISO/IEC 9798-2-6	1	✓				0.57	3.74	18.42	67.01	196.30	488.04	21278.58
						0.05	0.04	0.05	0.05	0.05	0.05	0.05
ISO/IEC 9798-3-6-1	2		✓		✓	43.08	802.95	8903.70	ME	ME	ME	ME
						0.13	0.18	0.19	0.19	0.19	0.19	0.19
ISO/IEC 9798-3-6-2	1		✓		✓	2.74	8.67	19.56	33.91	52.51	69.48	90.04
						0.12	0.15	0.15	0.15	0.15	0.15	0.15
ISO/IEC 9798-3-7-1	2		✓		✓	40.43	740.47	7483.36	16631.42	ME	ME	ME
						0.13	0.18	0.19	0.19	0.19	0.19	0.19
ISO/IEC 9798-3-7-2	1		✓		✓	2.38	7.71	16.68	26.99	35.06	49.49	TO
						0.22	0.32	0.33	0.33	0.33	0.33	0.33
PANA-AKA	5	✓	✓	✓	✓	5769.53	TO	TO	TO	TO	TO	TO
						0.10	0.10	0.10	0.10	0.10	0.10	0.10

Table 2. Experimental results. The time is in seconds. **No**: Number of abstractions. Properties of interest are **S**ecrecy, **A**liveness, **W**eak agreement, and **N**on-injective agreement.

H.2 Avantssar tools

The AVANTSSAR platform is an integrated toolset for the formal specification and **A**utomated **VA**lidatio**N** of **T**rust and **S**ecurity of **S**ervice-oriented **AR**chitectures. It provides three validation back-ends (CL-Atse, OFMC, and SATMC) which share the input languages for specifying protocols. The validators are based on two different techniques. SATMC reduces protocol insecurity problems to the satisfiability of propositional formulas which can then be checked by modern SAT solvers. CL-Atse and OFMC both use constraint solving techniques to search for attacks. However, they use different optimization strategies to reduce the search space. All these tools can verify protocols only for a bounded number of threads.

We have experimented with CL-Atse (version 2.5-21), OFMC (version 2013b), and SATMC (version 3.4) on several protocols from IKE and ISO/IEC 9798 families. Moreover, we have performed experiments on variants of the TLS and basic Kerberos protocols. For TLS, we distinguish two instances according to different security properties of interest. So far, we have not modelled IKE protocols for SATMC, as this requires substantial effort to encode oracles for Diffie-Hellman equations. We therefore defer extended experiments with SATMC to future work.

In our experiments, we measure the verification time for different numbers of sessions. Note that a session in CL-Atse, SATMC, and OFMC differs from a thread in Scyther. CL-Atse and SATMC specify a session as an instantiation of all protocol roles, not just a single role. For instance, a session of a protocol with three different roles results in three role instances (or three threads in Scyther) where a concrete agent is assigned to each role. In contrast, OFMC works with symbolic sessions where the agents executing the roles are not concretely specified but kept as variables.

For the AVANTSSAR tools, our experimental results generally exhibit smaller speedups than for Scyther. There is also a considerable variance between the different tools.

CL-Atse (Table 3) CL-Atse shows minor performance gains for the two IKEv1 protocols (pk2-a and pk-a2). However, abstraction enables the verifications of first three IKEv2 protocols (eap, eap2, and mac) for four sessions in less than 2 hours and dramatically speeds up the verification of three sessions of the eap and eap2 variants by factors greater than 690 and 900, respectively. For the last two IKEv2 protocols, the performance gains are still substantial: for four sessions we achieve a speedup factor of 7 for IKEv2-mactosig and of 107 for IKEv2-sigtomac. The best result in the ISO/IEC family is achieved for the ISO/IEC 9798-2-5 protocol where we can turn a timeout for 10 sessions into a time less than 0.2 seconds. The speedup for the 2-5 variant is less impressive and for the two 3-7 variants, we even observe an increase in verification time as we do for the basic Kerberos protocol. For TLS, the verification time of secrecy up to five sessions drops from 260 minutes to 6 minutes (factor 42), whereas that of authentication is sped up by a factor of 1.5 for four sessions.

OFMC (Table 4) Surprisingly, the experimental results for OFMC are almost dual to those for CL-Atse. In particular, for the two IKEv1 protocols, OFMC

loses performance on the abstracted protocols compared to the originals. Nevertheless, the abstractions save a lot of effort for the remaining protocols. We are able to increase the number of tractable sessions for 8 protocols: for 2 out of 7 from the IKE family, 5 out of 6 from the ISO/IEC 9798 family, and for the basic Kerberos protocol. For TLS, the verification of authentication is 1.7 times faster (up to 3 sessions). For secrecy, the tool achieves a 20-fold speedup (up to 4 sessions). As a typical case, OFMC verifies an abstraction of ISO/IEC 9798-2-5 for 5 sessions within less than 4 seconds whereas it times out on the original for more than 2 sessions.

SATMC (Table 5) The abstractions enable the verification of the Kerberos and TLS protocols for 5 and even 10 sessions. In particular, the tool takes less than 21 seconds to verify the abstracted TLS protocol for 10 sessions whereas it times out for 5 sessions of the original protocol. On the negative side, SATMC loses performance for the protocols in the ISO/IEC family.

Apart from positive results, our experiments also provide an evidence that protocol abstractions are not always helpful. This is typically the case when an abstraction removes sensitive information. In particular, the performance degradation for the AVANTSSAR tools can possibly be attributed to an interference with the highly refined optimization techniques used in these tools. More precisely, an abstraction may get rid of data that is crucial to eliminate redundancies (for CL-Atse) or to limit the number of branching nodes in the symbolic search tree (for OFMC). As a result, the search space becomes larger in the abstracted protocols than in the originals. However, the influence of abstraction on the SATMC's performance is not clear. A further investigation is therefore desirable.

H.3 ProVerif tool

ProVerif is an automated cryptographic protocol verifier in the standard Dolev-Yao model. It supports user-defined equational theories to model algebraic properties of cryptographic primitives. In constrast with Scyther, it uses approximations, e.g., translating protocol models in the applied pi calculus to a set of Horn clauses, to handle an unbounded number of sessions. These approximations are sound with respect to attacks, i.e., if the tool finds no attacks then the protocol is indeed secure.

We have validated our abstractions for ProVerif (version 1.88) on six protocols from the IKE and ISO/IEC 9798 families (see Table 6). For all these protocols, we observe good speedups. In particular, for the IKEv1-pk-a2 and the IKEv2-eap, the speedup factors are 6 and 5, respectively. The performance gains for the ISO/IEC 9798 protocols are less obvious than for the IKE ones. Concretely, the tool is roughly 1.5 times faster for these protocols.

protocol/prop./#sessions	S	A	W	N	3	4	5	10
IKEv1-pk2-a	✓			✓	0.06 0.05	0.11 0.08	0.53 0.32	TO TO
IKEv1-pk-a2	✓			✓	0.05 0.05	0.09 0.07	1.79 1.17	TO TO
IKEv2-eap	✓			✓	625.75 23.17	TO TO	TO TO	TO TO
IKEv2-eap2	✓			✓	1248.57 37.93	TO TO	TO TO	TO TO
IKEv2-mac	✓			✓	2.78 0.89	TO 5830.38	TO TO	TO TO
IKEv2-mactosig	✓			✓	0.24 0.12	1056.31 149.19	TO TO	TO TO
IKEv2-sigtomac	✓			✓	2.52 0.10	16710.31 155.63	TO TO	TO TO
ISO/IEC 9798-2-5	✓				20.05 0.52	TO 4064.93	TO 0.18	TO 0.17
ISO/IEC 9798-2-6	✓				1639.32 703.55	TO TO	TO TO	TO TO
ISO/IEC 9798-3-7-1		✓		✓	1.21 1973.78	4495.43 TO	TO TO	TO TO
ISO/IEC 9798-3-7-2		✓		✓	29.95 TO	TO TO	TO TO	TO TO
Kerb-basic			✓		0.30 0.18	0.29 0.18	22473.21 TO	TO TO
TLS-auth				✓	0.10 0.08	60.02 39.42	TO TO	TO TO
TLS-sec				✓	0.07 0.05	8.63 0.51	15551.63 369.57	TO TO

Table 3. Experimental verification results for CL-Atse. The time is in seconds.

protocol/prop./#sessions	S	A	W	N	2	3	4	5
IKEv1-pk2-a	✓			✓	36.28 59.10	27745.29 TO	TO TO	TO TO
IKEv1-pk-a2	✓			✓	4.28 12.09	849.46 9192.14	TO TO	TO TO
IKEv2-eap	✓			✓	8920.57 10.07	TO 8942.94	TO TO	TO TO
IKEv2-eap2	✓			✓	5407.00 46.14	TO TO	TO TO	TO TO
IKEv2-mac	✓			✓	18.59 11.19	22547.87 16139.98	TO TO	TO TO
IKEv2-mactosig	✓			✓	22.08 9.27	15561.69 10605.58	TO 11782.39	TO TO
IKEv2-sigtomac	✓			✓	18.58 12.36	13617.91 12408.54	TO TO	TO TO
ISO/IEC 9798-2-5	✓				805.64 3.61	TO 3.43	TO 3.85	TO 3.59
ISO/IEC 9798-2-6	✓				7232.17 144.06	TO TO	TO TO	TO TO
ISO/IEC 9798-3-6-1		✓		✓	17941.80 27.92	TO 18019.32	TO TO	TO TO
ISO/IEC 9798-3-6-2		✓		✓	TO 12.97	TO 3673.20	TO TO	TO TO
ISO/IEC 9798-3-7-1		✓		✓	TO 50.52	TO TO	TO TO	TO TO
ISO/IEC 9798-3-7-2		✓		✓	TO 11.61	TO 4010.64	TO TO	TO TO
Kerb-basic			✓		20.63 8.07	TO 28699.72	TO TO	TO TO
TLS-auth				✓	9.12 8.88	6002.38 3549.25	TO TO	TO TO
TLS-sec				✓	0.27 0.15	13.62 1.97	1304.21 59.87	TO TO

Table 4. Experimental verification results for OFMC. The time is in seconds.

number of sessions	S	A	W	N	3	4	5	10
ISO/IEC 9798-2-5	✓				0.44 0.58	0.42 0.64	0.45 0.90	0.50 3.70
ISO/IEC 9798-2-6	✓				0.45 35.36	0.46 247.67	0.48 2155.28	0.50 23740.06
ISO/IEC 9798-3-7-1		✓		✓	0.46 0.78	0.47 0.95	0.48 1.31	0.53 8.17
ISO/IEC 9798-3-7-2		✓		✓	0.47 2.64	0.47 5.83	0.64 11.61	0.60 121.17
Kerb-basic			✓		100.88 3.32	107.66 3.46	ME 51.15	TO 23396.15
TLS-auth				✓	163.51 1.52	4464.73 1.90	TO 2.65	TO 20.74
TLS-sec				✓	148.21 1.71	4002.34 1.90	TO 2.30	TO 8.85

Table 5. Experimental verification results for SATMC. The time is in seconds.

protocol/prop./#threads	S	A	W	N	∞
IKEv1-pk2-a	✓			✓	43.53 15.11
IKEv1-pk-a2	⋆				1.84 0.3
IKEv2-eap	✓			✓	22.27 4.22
IKEv2-mactosig	✓			✓	4.57 0.91
ISO/IEC 9798-2-5	✓				0.09 0.06
ISO/IEC 9798-3-7-1				✓	0.13 0.08

Table 6. Experimental verification results for ProVerif. The time is in seconds. The ⋆ presents ProVerif verifies the property for one role and cannot prove it for the other.