A Decision Procedure for Solving Constraint Systems in Presence of Multiple Independent Intruders

Ali Kassem, Pascal Lafourcade and Yassine Lakhnech
Verimag, Grenoble University, France
Firstname.Lastname@imag.fr

Sebastian Mödersheim
DTU Compute
Technical University of Denmark
samo@dtu.dk

***Abstract*—We consider a model of multiple independent intruders that have no ability to share knowledge between each other. We use this model to analyze security in wireless ad-hoc networks, where each intruder has a local control in the network, i.e., he can read and send messages only to his direct neighbors. Another application is the mobile ambient calculus where several intruder processes are not able to exchange their knowledge. Both these security problems can be reduced to satisfiability of lazy intruder constraint systems, for a bounded number of steps of the honest agents. However, the constraint-based verification method usually relies on a well-formedness property of constraints. This well-formedness entails that the constraints can be ordered so that the intruder knowledge is monotonically growing. This does not hold for several intruders that learn independent of each other. For the resulting generalized class of *weak-well-formed* constraints, we give a novel constraint reduction procedure and prove that it is sound, complete and terminating. We also prove that it is NP-complete.**

***Keywords*- Multiple independent intruders, constraint-based deduction, routing protocols, mobile code.**

I. Introduction

It is common in security to model one single intruder who works against all other honest participants. Even if we have in practice more than one dishonest adversary, the worst case is that they all collaborate. This allows one to simply think of all dishonest participants to be agents under control of one intruder. Moreover, one usually considers that the entire communication medium is controlled by the intruder: he can see all messages sent, block messages, insert any message he knows claiming any sender name. Even though it is of course unrealistic that an intruder controls the entire Internet, it makes sense again as a worst case assumption, because any communication line that is not secured physically should be assumed insecure and therefore messages must be protected using cryptography. This gives us a simple and powerful intruder model, namely an intruder that basically *is* the communication medium: every messages sent by an honest agent goes directly into the intruder knowledge and every message received by an honest agent is something from the intruder knowledge.

The conglomeration of all the dishonest participants into one single intruder however relies (besides their willingness to collaborate) on the assumption that all these participants can arbitrarily exchange messages, i.e., whenever one of them learns any value he can immediately tell it to every one of the intruders—so that we can truly work with *one* global intruder knowledge. This assumption obviously holds true when all communication lines are controlled by the intruders, but it does not, if we consider a world where intruders may be to some degree *isolated* from each other.

In this paper, we consider a model of several independent intruders that are willing to collaborate, but that cannot directly communicate with each other. We give two scenarios where such an intruder model is relevant: routing in wireless ad-hoc networks and the mobile ambient problem.

Wireless ad-hoc networks are decentralized networks, they do not rely on a pre-existing infrastructure. They crucially depend on efficient and correct *routing* protocols. Routing protocols aim at establishing a *valid route* between a source S and a destination D, i.e., a route representing an existing path in the network from S to D where every two adjacent nodes on the route are indeed real neighbors in the network. We speak of a successful attack on a routing protocol when the protocol establishes an invalid route. For a model where all dishonest nodes collaborate and can freely communicate with each other, Arnaud et. al. show that security of routing protocols is co-NP-complete for a bounded number of sessions [ACD10]. However, many routing protocols are trivially vulnerable to such a powerful intruder who can launch *wormhole attacks*, i.e., direct communication between physically distant dishonest nodes [HPJ06], [LPM$^+$05]. In contrast many routing protocols are secure in a *weaker* intruder model. In the model we consider, only neighboring dishonest nodes can directly communicate. This weaker model is in fact more realistic than the powerful intruder: Especially in wireless ad-hoc setting, the intruder may have only a limited number of devices under his control, and each with a limited range. An extreme example is the case of ad-hoc sensors that are thrown from a plane into the enemy field during a battle.

The second scenario of multiple independent intruders that we consider is a problem of mobile ambients: an intruder has written some malicious code that is being executed on one or more honest platforms, e.g., a web-browser, mobile phone, or in a virtual machine. Each piece of code can be

thought of as one independent intruder, and a priori, these intruders cannot communicate with each other: a platform may have the code compute on some secret and try to ensure that this secret never leaves the platform.

This model of independent intruders however clashes with one of the most successful protocol verification techniques: the constraint-based method of protocol verification, that we simply call *lazy intruder* for short [Hui99], [MS01], [CKRT03], [BMV05]. The idea of the lazy intruder is to avoid exploring the space of all messages the intruder can construct each time he is sending a message to an honest agent. Instead we note a constraint $T \vdash x$ where x is a new variable and T is the set of messages he currently knows and $\vdash$ represents the intruder deduction relation, i.e., whatever x is, it must be something that can be deduced from knowledge T. The variables like x get then instantiated in a demand-driven, lazy way: we instantiate them only when it is necessary for performing some transition. Note that the answer that an honest agent sends in reply may also contain variables that occurred in messages he received previously. Therefore the intruder knowledge T' at a later point can contain variables. Note that in the normal case of a single intruder, we can rely on a *well-formedness* property of the constraints: if knowledge T' contains a variable x, then there must be an "earlier" constraint $T \vdash t$ where x occurs in t and $T \subseteq T'$ because the intruder knowledge is growing monotonically. This is very helpful in the checking the satisfiability of constraints; in a nutshell, once the $T \vdash t$ constraint is solved we can rely that x is something the intruder could construct from knowledge T, thus it is subsumed by the knowledge T' and we then do not need to analyze it. It is well-known that the satisfiability problem (and thus insecurity for a bounded number of sessions) is NP-complete in the free algebra. Moreover there exist several extensions for some algebraic theories like for instance in [RT03], [CLS03], [MS03], [CKRT03], [DLLT08], [CLCZ10].

In the case of multiple intruders, however, the monotonicity does not hold anymore. Here it may happen that $T \vdash x$ represents that one of the intruders with knowledge T said some message x to an honest agent, and later that agent says a message that contains x to another intruder who has knowledge T'. If these intruders have been working independently, $T \subseteq T'$ may not hold.

It it thus a question whether one can adapt the lazy intruder technique to handle also constraints without the well-formedness assumption and so to apply this efficient technique to the scenarios we have described. Indeed, Avanesov et al. [ACRT10], [ACRT12] show that even without well-formedness, satisfiability of the intruder constraints is NP-complete. Note that the detailed description of the method and the proof of its correctness spans around 40 pages. Even more crucially, the arguments rely on the (non-deterministic) exploration of all possible solutions up to a certain (polynomial) size. This defies the basic idea of the lazy intruder to avoid the complete exploration of possible messages the intruder could say, and is thus primarily of conceptual value.

Contributions: We present a novel reduction calculus for the satisfiability of lazy intruder constraints without well-formedness that arise in the multiple-intruder setting. The calculus is declarative and conceptually simple, and so are the formal proofs of its soundness, completeness, and termination. Also the proof of NP-completeness of the problem, verifying [ACRT10], [ACRT12], is almost immediate in our formalization.

Further, our calculus is close in its spirit to the original lazy intruder technique, lazily narrowing to the possible solutions. The main change to handle non-well-formed constraints is as follows: if the variable x represents an arbitrary message than one intruder has created, and x later appears in the knowledge of another intruder, then our calculus will "optimistically" assume that the first intruder transmits its entire knowledge via x to the second intruder—as long as the reductions do not force us to instantiate x in a particular way. Only in this case, we check that our optimistic derivations are still possible under the different instantiation of x. Thus we also handle the non-monotonic knowledge problem in a truly demand-driven, lazy way. We claim that with this idea in fact all the various lazy intruder results for different algebraic theories can be generalized to work with non-well-formed constraints.

Further, we apply the multiple lazy intruder technique in two applications. First, we formalize routing protocols for wireless ad-hoc networks assuming independent dishonest nodes. Here we give an example of a modified version of the Secure Routing Protocol SRP [PH02] applied to the Dynamic Source Routing protocol DSR [JMB01]. This protocol has an attack in the standard intruder model where all dishonest nodes can communicate with each other directly, but is safe in our more realistic model of independent intruders (as long as there is an honest node on every path between source and destination). Second, we show how to extend the lazy mobile intruder approach of [MNN13] to communication with arbitrary cryptographic messages which also produces non-well-formed constraints.

Outline: In Section II, we introduce definitions and notations. In Section III, we summarize standard lazy intruder procedure for well-formed constrains. Then in Section IV, we first define the form of constraint system for weak-well-formed constraints, i.e., for the case of multiple intruders. Then we design some rules and prove that our procedure is terminating, sound, and complete. We also prove that it is NP-complete. Finally before concluding we present two applications in Section V. First we model routing protocols and show how to reducing security properties for a bounded number of sessions to satisfiability of *weak-well-formed* constraints. Then, we extend the result of [MNN13] in order

to obtain a conjunction of weak-well-formed constraints for ambient calculus. Note that missing proofs are given in Appendix.

II. Preliminaries

Messages are represented by terms build over the signature given in Figure 1. We consider a countable set of variables $\mathcal{V}$. For a set $V \subseteq \mathcal{V}$, we define a set of terms $\mathcal{T}(\mathcal{F}, V)$ to be the smallest set containing $\mathcal{F}_0$ and V, such that for $f \in \mathcal{F}_n$: if $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, V)$ then $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, V)$. We do not consider any equational theory then in the rest of the paper we use the standard syntactic subterms. In the case that $V = \emptyset$, we simply write $\mathcal{T}(\mathcal{F})$, this is the set of ground terms. For a term t, let $vars(t)$ be the set of variables that occur in t, such that for some $f \in \mathcal{F}_n$:

$$vars(t) = \begin{cases} \{t\} & \text{if } t \in \mathcal{V} \\ vars(t_1) \cup \cdots \cup vars(t_n) & \text{if } t = f(t_1, \ldots, t_n). \end{cases}$$

$inv(\cdot)$	private key of a given public key
$scrypt_{\cdot}(\cdot)$	symmetric encryption
$hmac_{\cdot}(\cdot)$	message authentication code
$acrypt_{\cdot}(\cdot)$	asymmetric encryption
$sig_{\cdot}(\cdot)$	digital signature
$\ll\cdot,\cdot\gg$	concatenation
$[]$	empty list
$\cdot :: \cdot$	list constructor
req	constant identifies the request phase
rep	constant identifies the response phase
i_k	name of intruder k
Ω	a constant available for all intruders
in,out, $open$	constructors used for ambient calculus

Figure 1. Signature $\mathcal{F}$.

Every intruder k has only access to his name i_k and also to all symbols of Figure 1 except $inv(\cdot)$ symbol. The functions that intruder has access to are called *intrudable* functions. Note that we use agent names as public keys.

Substitutions and Unifications: A *substitution* σ is a mapping from $\mathcal{V}$ to $\mathcal{T}(\mathcal{F}, \mathcal{V})$ with the domain $dom(\sigma) = \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$. We say that a substitution σ is *ground*, if for every $x \in dom(\sigma)$, the term $\sigma(x)$ is ground. We extend σ to a homomorphism on terms and set of terms as expected. We say that the two terms t and s are *unifiable*, if there exists a substitution θ, called *unifier*, such that $\theta(t) = \theta(s)$. We define the *most general unifier*, or for short mgu, of two terms t and s to be a unifier, denoted $mgu(t, s)$, such that for any unifier θ of t and s, there exists a substitution σ such that $\theta = \sigma \circ mgu(t, s)$, where $\circ$ is a composite of two mappings.

III. The Conventional Lazy Intruder

We define how the intruder can deduce new messages from a given set of messages that he has initially and that he observed by eavesdropping on the network. Namely, how he can decompose and compose messages in order to build new ones. We model the ability of the intruder by the deduction relation $\vdash$. The relation $T \vdash t$ means that the term t is deducible from the set of terms T. In Figure 2, we define this relation with a deduction system. The composition rule (C) expresses that, for every intrudable function $f \in \mathcal{F}_n$, and for any terms $t_1, \ldots, t_n$ that the intruder can derive, he can compose the term $f(t_1, \ldots, t_n)$, so that he can compose messages by pairing, building lists, encrypting and signing messages providing he has the corresponding keys. The rules (UP) and (UL) express respectively that the intruder can decompose a pair and a list into their components. The axiom rule (A) expresses that the given message t is contained on the set T. The rule (SD) expresses that the intruder can decrypt symmetric cipher only if he knows the encryption key. The last two rules (AD) and (US) express respectively that each term encrypted by a public key t_2 can be decrypted by the intruder only if he knows the corresponding private key $inv(t_2)$, the same he can read a term signed by the private key $inv(t_2)$ provided that he knows the corresponding public key t_2. Note that $inv(inv(t)) \neq t$ and $inv(\cdot)$ is not intrudable: the intruder cannot obtain the private key from a known public key.

A. Syntax and Semantics of Constraints

We use constraints ϕ, ψ etc. over the following language:

$$\begin{array}{rcl} \phi, \psi & ::= & T \vdash t \\ & | & \phi \wedge \psi \\ & | & x = t \end{array}$$

where x is a variable, t is a term and T is a set of terms.

Let $\mathcal{I}$ be a mapping from variables to ground terms, extended to a morphism on messages and sets of messages as expected. Then we define:

$$\begin{array}{lll} \mathcal{I} \models T \vdash t & \text{iff} & \mathcal{I}(T) \vdash \mathcal{I}(t) \\ \mathcal{I} \models \phi \wedge \psi & \text{iff} & \mathcal{I} \models \phi \text{ and } \mathcal{I} \models \psi \\ \mathcal{I} \models x = t & \text{iff} & \mathcal{I}(x) = \mathcal{I}(t) \end{array}$$

The conventional lazy intruder is based on the following assumption of *well-formedness*:

Definition III.1 (Well-formedness)**.** *Given a constraint $\phi \equiv T_1 \vdash t_1 \wedge \ldots \wedge T_n \vdash t_n \wedge \phi_0$, where ϕ_0 is a conjunction of equality constraints. We say that ϕ (in this ordering of the $T_i \vdash t_i$) satisfies the* knowledge monotonicity *property iff $T_i \subseteq T_j$ whenever $1 \leq i \leq j \leq n$. We say that ϕ satisfies the* variable origination *property iff for each variable $x \in vars(T_j)$ (with $1 \leq j \leq n$) follows $x \in vars(t_i)$ for some $1 \leq i \leq j$. A constraint ϕ is called* well-formed *iff its deduction conjunctions can be ordered so that both the* variable origination *and* knowledge monotonicity property *hold.*

Intuitively, the ordering of the constraints is the temporal order (i.e., in which the t_i have been sent by the intruder), knowledge monotonicity means that the intruder knowledge

$$(C)\ \frac{T \vdash t_1 \cdots T \vdash t_n}{T \vdash f(t_1, \ldots, t_n)}\ f \text{ is intrudable} \qquad (UP)\ \frac{T \vdash \ll t_1, t_2 \gg}{T \vdash t_i}\ i \in \{1,2\} \qquad (UL)\ \frac{T \vdash t_1 :: t_2}{T \vdash t_i}\ i \in \{1,2\} \qquad (A)\ \frac{t \in T}{T \vdash t}$$

$$(SD)\ \frac{T \vdash scrypt_{t_2}(t_1)\ \ T \vdash t_2}{T \vdash t_1} \qquad (AD)\ \frac{T \vdash acrypt_{t_2}(t_1)\ \ T \vdash inv(t_2)}{T \vdash t_1} \qquad (US)\ \frac{T \vdash sig_{inv(t_2)}(t_1)\ \ T \vdash t_2}{T \vdash t_1}$$

Figure 2. Intruder deduction system.

can only grow over time, and variable origination means that variables do come out of the blue but first occur in a term t_i that the intruder sends. The assumption is crucial for the completeness of the conventional lazy intruder procedure.

In the following section we will relax this well-formedness assumption (and calling it *weak well-formedness*) by dropping the knowledge monotonicity property—to allow for multiple intruders that can learn independently, so their knowledges cannot necessarily be ordered like this. We define that a constraint is *simple* (or *in solved form*) if for every conjunct $T \vdash t$ it holds that $t \in \mathcal{V}$.

B. A Proof Calculus

The lazy intruder proof calculus is a declarative way to describe much of the constraint reduction and to keep it separated from some of the technical aspects. The proof rules have the form $\frac{\psi}{\phi}$ and should be read as follows: if constraint ψ is satisfiable, then so is ϕ. One would use them backwards, i.e., starting with a constraint ϕ, find all applicable rules to reduce to some simpler ψ—until a *simple* form (*solved* form) has been found. We consider only the 3 following proof rules in the free algebra:

Generate Rule: :

$$\frac{T \vdash t_1 \wedge \ldots \wedge T \vdash t_n \wedge \phi}{T \vdash f(t_1, \ldots, t_n) \wedge \phi}$$

for any operation symbol f that is available to the intruder.

Unification Rule:

$$\frac{\sigma(\phi) \wedge eq(\sigma)}{T \vdash t \wedge \phi}\ s, t \notin \mathcal{V}, s \in T, \sigma \in mgu(s,t)$$

where $mgu(s,t)$ denotes the set of most general unifiers of s and t and $eq([x_1 \mapsto s_1, \ldots, x_n \mapsto s_n])$ is the formula $x_1 = s_1 \wedge \ldots \wedge x_n = s_n$.

Analysis Rule: The analysis rule for asymmetric encryption for instance looks as follows, allowing the intruder to obtain the plain-text of a message but adding the constraint that he knows the respective private key:

$$\frac{T \vdash inv(k) \wedge (T \vdash t \wedge \phi)^{m \gg T}}{T \vdash t \wedge \phi}\ acrypt_k(m) \in T$$

where $\phi^{m \gg T}$ shall denote that we look in ϕ for all conjuncts of the form $T' \vdash m'$ and if $T \subseteq T'$, we replace it with $T' \cup \{m\} \vdash m'$, i.e., in all super-knowledges of T the message m will be available. This update of the intruder knowledge is convenient to preserve well-formedness without any cumbersome constructions.

For opening a signature, the intruder needs the corresponding public key:

$$\frac{T \vdash k \wedge (T \vdash t \wedge \phi)^{m \gg T}}{T \vdash t \wedge \phi}\ sig_{inv(k)}(m) \in T$$

For symmetric encryption, the intruder needs the key itself:

$$\frac{T \vdash k \wedge (T \vdash t \wedge \phi)^{m \gg T}}{T \vdash t \wedge \phi}\ scrypt_k(m) \in T$$

Finally, from pairs and lists he gets the components without any requirements on his knowledge:

$$\frac{(T \vdash t \wedge \phi)^{m_1, m_2 \gg T}}{T \vdash t \wedge \phi}\ \ll m_1, m_2 \gg \in T \text{ or } m_1 :: m_2 \in T$$

IV. Multiple Lazy Intruders

The "classical lazy intruder" of the previous section was based on the assumption of well-formedness, i.e., that all variables originate from intruder choices and the intruder knowledge grows monotonically. Therefore, in a constraint $T \vdash t$ with $x \in T$, we know there is an "earlier" constraint $T_0 \vdash t_0$ such that $T_0 \subseteq T$ and x occurs in t_0. Intuitively, x is part of a choice the intruder made earlier, at a smaller knowledge. Suppose we first apply reduction to $T_0 \vdash t_0$, then we either instantiate x with something more concrete, or we eventually end up with $T_0 \vdash x$. In this case, we know that whatever x is, it can be constructed from T_0. Thus, we can safely ignore the message x in $T \vdash t$ (some constraint systems even do remove it). In fact, this is the reason why the calculus is complete despite the fact that analysis and unification rules cannot be applied to a variable x in the knowledge.

We now like to consider multiple intruders that learn messages independent of each other and who are separated so that they cannot pool their knowledge. For instance, consider two intruders with knowledge $T_1 = \{k, c_1\}$ and $T_2 = \{k, c_2\}$ and between whom sits an honest agent who would be happy to forward one arbitrary message received from the first intruder to the second intruder. Finally, let the goal be that the second intruder can produce c_1. This can be expressed by the constraint: $T_1 \vdash x \wedge T_2 \cup \{x\} \vdash c_1$. This constraint is not well-formed, because $T_1 \subsetneq T_2$. On non-well-formed constraints, the classical lazy intruder is still sound and terminating, but not necessarily complete. For the given example, it would

miss the solution $x = c_1$ (because the unification rule cannot be applied between c_1 and x). We thus define a weaker version of the well-formedness assumption that is satisfied for constraints produced by multiple intruders:

Definition IV.1 (Weak well-formedness)**.** *A constraint ϕ is called* weakly well-formed *iff its deduction conjunctions can be ordered so that the variable origination property holds: $\phi \equiv T_1 \vdash t_1 \wedge \ldots \wedge T_n \vdash t_n \wedge \phi_0$, where ϕ_0 is a conjunction of equality constraints, and for every variable $x \in vars(T_i)$, $1 \leq i \leq n$, exists j, $1 \leq j < i$, such that $x \in vars(t_j)$. Moreover well-formedness requires that for every equation $x = t$, x cannot occur on the left-hand side of any other equation or in $T \vdash t$.*

Again, in applications the order on the constraints is simply the temporal order in which the constraints arise, and weak well-formedness means that variables have to first occur in a term t_i that an intruder says. Therefore, every variable x that occurs in the constraints is associated to one conjunct $T_i \vdash t_i$ in which it first occurs (in t_i) and the possible values of x "originating" from the knowledge T_i (although in general $x \in T_i$ does not hold). For a well-formed constraint ϕ, let us thus denote with $know_\phi(x)$ this knowledge T_i from which x originates.[1] We may omit the subscript ϕ when clear from the context. The rest of this section is devoted to design a sound, complete, and terminating reduction procedure for weakly well-formed constraints. The main idea for this can be illustrated by the above example: here we could use the variable x as a kind of "channel" through which the first intruder could transmit all his current knowledge to the second intruder, i.e., he could simply choose $x = \ll k, c_1 \gg$. This channel idea does, however, not work in general: we may have further constraints on x that destroy the channel. Suppose we have the further constraint $\{f(x)\} \vdash f(scrypt_k(y))$ then there is still a solution, namely $y = c_1$ and $x = scrypt_k(y)$, but the simple solution of choosing x to be the concatenation of the entire knowledge T_1 does not work then.

In fact, we thus use an "optimistic strategy": as long as we are not forced to instantiate variables in a particular way, we just assume that they can transport the entire knowledge from which they were created. We would that replace the above constraint $T_2 \cup \{x\} \vdash c_1$ with $T_1 \cup T_2 \vdash c_1$ optimistically. We have to remember however that this is optimistic and relies on variable x to transport the entire knowledge, so we need to remember the original constraint (modulo instantiations of variables we have made) so we can revert to it when x gets instantiated. This motivates a new form of constraints to carry the entire information.

[1] In some cases, this knowledge is not uniquely determined because there may be more than one order of the $T \vdash t$ conjuncts such that the variable origination property is satisfied. In this case, we simply assume that one fixes one such order throughout the constraint reduction.

A. Optimistic Constraint Systems

Definition IV.2 (Optimistic Constraint System)**.** *An* optimistic constraint system *has the form $\phi[V]\phi'$ where ϕ and ϕ' are conventional weakly well-formed intruder constraints and V is a set of variables. For every equation $x = t$ of ϕ and ϕ', $x \notin V$.*

The idea is that ϕ is the original constraint (modulo instantiations of variables we always perform throughout ϕ and ϕ'), and ϕ' is what is left to solve if we can use all variables in V as channels, i.e., if we can instantiate them arbitrarily). In fact we treat both $[V]$ and ϕ' as an *annotation* to the constraint ϕ.

Definition IV.3 (Semantics)**.** *An interpretation $\mathcal{I}$ (mapping all variables to ground terms) satisfies an optimistic constraint system if it satisfies the classical part: $\mathcal{I} \models \phi[V]\phi'$ iff $\mathcal{I} \models \phi$. The meaning of the annotation $[V]\phi'$ is determined only indirectly by an invariant. Let ϕ and ϕ' be weakly well-formed. Let $R_V^{\phi'} \equiv \bigwedge_{x \in V} \{x\} \vdash know_{\phi'}(x)$. We omit ϕ' in $R_V^{\phi'}$ when it is clear from the context. We define: $invariant(\phi[V]\phi')$ iff $\phi' \wedge R_V^{\phi'} \models \phi$. As is standard, the $\models$ relation between formulae here denotes logical implication, i.e., all models of the first formula are models of the second.*

The invariant thus expresses the following property our procedure will rely on: for solving ϕ it is sufficient to solve ϕ' as long as every variable $x \in V$ can be instantiated to communicate the entire knowledge $know_{\phi'}(x)$—so intruders who know x also know $know_{\phi'}(x)$. The invariant ensures that in all interpretations in which this holds, also ϕ holds.

We prove below that all constraint reductions we consider in this paper preserve the invariant, and we will start with the constraint $\phi[]\phi$ for which the invariant trivially holds, since $\phi \models \phi$. In contrast, it would be difficult to check for an arbitrary optimistic constraint store whether it satisfies the invariant. We therefore define the notion of simple constraints independent of the invariant.

Definition IV.4 (Simplicity)**.** *An optimistic constraint store is* simple *if it has the form $\phi[V]\phi'$ and ϕ' is simple.*

Lemma IV.1 (Satisfiability)**.** *If $\phi[V]\phi'$ is simple and satisfies the invariant, then ϕ is satisfiable.*

Proof: Let $\phi[V]\phi'$ be a simple constraint store, then ϕ' is simple. Consider the following set of equations:

- Every $x = t$ that occurs in ϕ'
- $x = \ll t_1, \ldots, t_n \gg$ for every $x \in V$ and $know_{\phi'}(x) = \{t_1, \ldots, t_n\}$ (choosing any order for the t_i).
- $x = \Omega$ for all other variables.

By weak well-formedness of ϕ', every variable x occurs only on the left-hand side of one unique equation in this system. We can thus, by successive replacement, obtain an interpretation $\mathcal{I}$ that is a model of this set of equations. This

interpretation satisfies ϕ' because every intruder knows (at any knowledge) the special constant Ω. Further $\mathcal{I} \models R_V$, because each $x \in V$ is instantiated with a concatenation of the terms in the respective knowledge. By the invariant follows $\mathcal{I} \models \phi$. Thus, we have constructed a model of ϕ and thus $\phi[V]\phi'$ is satisfiable. ■

B. Expansion and Substitution

A key element of optimistic constraint stores is that we work temporally under the assumption that every variable x in the intruder knowledge carries the entire knowledge it originates from, i.e., $know_{\phi'}(x)$. We thus define a function that expands the intruder knowledges in ϕ' as follows:

$$\mathsf{expand}(\phi[V]\phi') = \begin{cases} \mathsf{expand}(\phi[\{x\} \cup V]T \setminus \{x\} \cup know_{\phi'}(x) \vdash t \wedge \phi'_0) \\ \qquad \text{iff } \phi' = \phi'_0 \wedge T \vdash t \text{ and } x \in T \cap \mathcal{V}, \\ \phi[V]\phi', \text{otherwise} \end{cases}$$

Note that in the first clause, there is a non-deterministic choice of the conjunct and variable to process first; the final result is however uniquely defined (i.e., every order of expansions leads to the same normal form where $T \cap \mathcal{V} = \emptyset$ for every conjunct $T \vdash t$ of ϕ').

Based on the definition of the expansion function, we define how to apply a substitution to the optimistic constraint system, recording also the substituted variables in an equation both on the ϕ and ϕ' side: $\sigma(\phi[V]\phi') =$

$$\begin{cases} \mathsf{expand}(\sigma(\phi) \wedge eq(\sigma)[]\sigma(\phi)eq(\sigma)) & \text{if } dom(\sigma) \cap V \neq \emptyset \\ \sigma(\phi) \wedge eq(\sigma)[V]\sigma(\phi') \wedge eq(\sigma) & \text{otherwise} \end{cases}$$

Lemma IV.2 (Closure)**.** *For every ϕ, the optimistic constraint $\phi[]\phi$ satisfies the invariant. Application of the function $expand$ and application of a substitution σ to $\phi[V]\phi'$ preserves the invariant and the weak well-formedness.*

C. Reduction Rules

For the new optimistic constraint stores, we now introduce reduction rules that are working in a similar way as the ones of the conventional lazy intruder.

Unify: First we have a unify rule that allows us to unify a term t to generate with a term s in the knowledge. We then apply the unifier σ to entire optimistic constraint store which, as defined above, will re-instantiate $\sigma(\phi)$ if σ substitutes any variable of V.

$$\frac{\sigma(\phi[V]\phi')}{\phi[V]T \vdash t \wedge \phi'} \; s \in T;\; s,t \notin \mathcal{V};\; \sigma \in mgu(s,t)$$

Generate: Next, we have a generate rule that reduces the derivation of a composed term $f(t_1, \ldots, t_n)$ to the derivation of its subterms t_i if f is an intrudable operation.

$$\frac{\phi[V]T \vdash t_1 \wedge \ldots T \vdash t_n \wedge \phi'}{\phi[V]T \vdash f(t_1, \ldots, t_n) \wedge \phi'} \; f \text{ intrudable}$$

Analyze Crypt: Finally, we have the analysis rules. For asymmetric encryption, when in a deduction constraint $T \vdash t$, T contains a term $acrypt_k(m)$, then we can try to derive the decryption key, i.e., add the constraint $T \vdash inv(k)$ and add the resulting m to every other constraint with knowledge that is a superset of T:

$$\frac{\mathsf{expand}(\phi[V]T \vdash inv(k) \wedge (T \vdash t \wedge \phi')^{m \gg T})}{\phi[V]T \vdash t \wedge \phi'} \; acrypt_k(m) \in T$$

Note that we apply here the $expand$ function to the resulting constraint. This is because the resulting term m of the analysis step may be a variable—in this case potentially being a channel from another intruder knowledge $know_{\phi'}(m)$. The other analysis rules are similarly adapted from the conventional lazy intruder.

We illustrate our procedure using the following constraint system ϕ. We have that $\mathcal{I}(x)$: $\mathcal{I}(x) = acrypt_k(k)$ and $\mathcal{I}(y) = k$ is a solution of ϕ.

$$\phi = \begin{cases} \{a, acrypt_k(k)\} \vdash x \\ \{x, inv(k), b\} \vdash scrypt_k(b) \\ \{f(x)\} \vdash f(acrypt_k(y)) \end{cases}$$

We show how to obtaine this solution by applying our procedure. We starte from $expand(\phi[]\phi)$:

$$\phi \quad [\{x\}] \quad \begin{cases} \{a, acrypt_k(k)\} \vdash x \\ \{a, acrypt_k(k), inv(k), b\} \vdash scrypt_k(b) \\ \{f(x)\} \vdash f(acrypt_k(y)) \end{cases}$$

Generate and *Unify* b on second:

$$\phi \quad [\{x\}] \quad \begin{cases} \{a, acrypt_k(k)\} \vdash x \\ \{a, acrypt_k(k), inv(k), b\} \vdash k \\ \{f(x)\} \vdash f(acrypt_k(y)) \end{cases}$$

Analysis on second:

$$\phi \quad [\{x\}] \quad \begin{cases} \{a, acrypt_k(k)\} \vdash x \\ \{a, acrypt_k(k), inv(k), b\} \vdash inv(k) \\ \{a, acrypt_k(k), inv(k), b, k\} \vdash k \\ \{f(x)\} \vdash f(acrypt_k(y)) \end{cases}$$

Unify $inv(k)$ on second and k on third:

$$\phi \quad [\{x\}] \quad \begin{cases} \{a, acrypt_k(k)\} \vdash x \\ \{f(x)\} \vdash f(acrypt_k(y)) \end{cases}$$

Unify on second: $\sigma = \{x \to acrypt_k(y)\}$: as $x \in dom(\sigma) \cap V$, the application of σ results in $expand(\sigma(\phi) \wedge eq(\sigma)[]\sigma(\phi) \wedge eq(\sigma))$.

$$\sigma(\phi) \wedge eq(\sigma) \quad [] \quad \begin{cases} \{a, acrypt_k(k)\} \vdash acrypt_k(y) \\ \{acrypt_k(y), inv(k), b\} \vdash scrypt_k(b) \\ \{f(acrypt_k(y))\} \vdash f(acrypt_k(y)) \\ x = acrypt_k(y) \end{cases}$$

Unify on first: $\sigma' = \{y \to k\}, \sigma'' = \sigma' \circ \sigma$:

$$\sigma''(\phi) \wedge eq(\sigma'') \quad [] \quad \begin{cases} \{acrypt_k(k), inv(k), b\} \vdash scrypt_k(b) \\ \{f(acrypt_k(k))\} \vdash f(acrypt_k(k)) \\ x = acrypt_k(y) \wedge y = k \end{cases}$$

Generate and *Unify* b on first:

$$\sigma''(\phi) \wedge eq(\sigma'') \quad [] \quad \begin{cases} \{acrypt_k(k), inv(k), b\} \vdash k \\ \{f(acrypt_k(k))\} \vdash f(acrypt_k(k)) \\ x = acrypt_k(y) \wedge y = k \end{cases}$$

Analysis $acrypt_k(k)$ on first:

$$\sigma''(\phi) \wedge eq(\sigma'') \quad [] \quad \begin{cases} \{acrypt_k(k), inv(k), b\} \vdash inv(k) \\ \{acrypt_k(k), inv(k), b, k\} \vdash k \\ \{f(acrypt_k(k))\} \vdash f(acrypt_k(k)) \\ x = acrypt_k(k) \wedge y = k \end{cases}$$

Unify on first, second and third:

$$\sigma''(\phi) \wedge eq(\sigma'') \quad [] \quad \{ x = acrypt_k(k) \wedge y = k$$

We now want to show that the reduction calculus is correct, complete and terminating, and since for a large part, the reduction is working on the annotation part $[V]\phi'$ of an optimistic constraint, we first need to prove that the invariants (and the weak well-formedness) are preserved by any application of the reduction rules.

D. Preservation of the Invariant

Unify Rule: Suppose the invariant holds for $\phi[V]T \vdash t \wedge \phi'$, $s \in T$, $s, t \notin \mathcal{V}$, $\sigma \in mgu(s,t)$. If $dom(\sigma) \cap \phi \neq \emptyset$, then the result is $\mathsf{expand}(\sigma(\phi) \wedge eq(\sigma) [] \sigma(\phi) eq(\sigma))$ for which the invariant and the weak well-formedness follows already by Lemma IV.2. Otherwise we have $\sigma(\phi) \wedge eq(\sigma)[V]\sigma(\phi') \wedge eq(\sigma)$. Now we have

$$\begin{aligned} & R_V \wedge \phi' \wedge eq(\sigma) \\ & \models R_V \wedge \phi' \wedge T \vdash t \quad \text{because } \sigma \in mgu(s,t) \\ & \models \phi \quad \text{by the invariant before reduction.} \end{aligned}$$

Weak well-formedness in this case: consider an order according to which $T \vdash t \wedge \phi'$ is well-formed. Let x be a variables that occurs in t and that originates in the $T \vdash t$ conjunct. The unifier σ then either

- substitutes x; and then the variables of $\sigma(x)$ originate in earlier constraints, or
- substitutes a variable y that occurs in $s \in T$ with a term containing x. Since y must originate in an earlier constraint $T_0 \vdash t_0$ in ϕ', we have that in $\sigma(\phi')$, x originates in $\sigma(T_0 \vdash t_0)$.

So in both cases variable origination of x or $\sigma(x)$ is preserved, even though the $T \vdash t$ constraint is removed.

Generate Rule: For the invariant, the derivation is again similar, this time using the implication: $\wedge_{i=1}^{n} T \vdash t_i \models T \vdash f(t_1, \ldots, t_n)$ if f is intrudable. Weak well-formedness is immediate if we consider the $T \vdash t_i$ (in any order) to take the position of $T \vdash f(t_1, \ldots, t_n)$ in the well-formedness order.

Analysis Rule: Let $acrypt_k(m) \in T$ and $T' \supseteq T$; then we have $T \vdash inv(k) \wedge (T' \cup \{m\}) \vdash t \models T' \vdash t$ thus $T \vdash inv(k) \wedge (\psi)^{m \gg T} \models \psi$ for any ψ. Then preservation of the invariant is immediate.

Weak well-formedness: the new conjunct $T \vdash inv(k)$ is in the order just before $T \vdash t$. The addition of m in T does not destroy the origination property because m is a subterm of a term in T.

E. Soundness

Lemma IV.3 (Soundness)**.** *All the constraint rules are sound, i.e., for every concrete reduction* $\frac{\chi'}{\chi}$ *with our calculus holds* $\chi' \models \chi$.

F. Completeness

The completeness lemma states that, if a constraint is satisfiable under a particular interpretation, then it is either already simple or there exists an applicable reduction rule that supports that interpretation. Together with soundness and termination below, this gives a decision procedure for satisfiability.

Lemma IV.4 (Completeness)**.** *Given a weakly well-formed conventional constraint* ϕ*. Starting with the optimistic constraint* $\mathsf{expand}(\phi[V]\phi)$*, all optimistic constraints that we can reach with the reduction rules are either simple, unsatisfiable, or admit the application of another reduction rule.*

G. Termination

Note that our calculus still admits infinite derivations: when an analysis rule is applicable, the same rule can be applied over and over again, but without changing the semantics of the constraint, since we just add a conjunct that we already have and extend the intruder knowledge of some constraints with a term that they already contain. Let us call an analysis step *redundant*, when it triggers the update $m \gg T$ for some $m \in T$. Note this is *not* a semantical check (like $\mathcal{I}(T) \vdash \mathcal{I}(t)$), but simply a syntactic measure to apply a rule again or for a trivial case. Excluding redundant steps, our calculus is terminating:

Lemma IV.5 (Termination)**.** *Given the optimistic constraint store* $\mathsf{expand}(\phi[]\phi)$ *for some weakly well-formed conventional constraint* ϕ*. Then the reduction rules do not admit an infinitely long sequence of non-redundant reduction steps.*

Theorem IV.1 (Decision Procedure)**.** *Given a weakly well-formed conventional intruder constraint* ϕ*. Then our extended calculus derive from* $\phi[]\phi$ *a finite set* $\{\phi_1[V_1]\phi_1', \ldots, \phi_n[V_n]\phi_n'\}$ *of optimistic constraints that are simple and have the same models as* ϕ*, i.e.,*

$$\mathcal{I} \models \phi \text{ iff } \mathcal{I} \models \phi_i[V_i]\phi_i' \text{ for some } i \in \{1, \ldots, n\} .$$

Note that ϕ is unsatisfiable iff $n = 0$.

The problem whether a weakly well-formed conventional constraint ϕ is satisfiable is NP-complete.

Proof: Compute all non-redundant reductions of $\mathsf{expand}(\phi[]\phi)$ to simple constraints. By the termination lemma, there is no infinitely long derivation. Moreover, the reduction relation is finitely branching, i.e., to an optimistic constraint system $\phi[V]\phi'$ every reduction rule can be applied only in finitely many ways. Thus by König's lemma, there are finitely many reachable optimistic constraint systems $\phi_i[V_i]\phi'_i$. By the soundness lemma, their models are a subset of the models of ϕ and by completeness lemma, a superset.

For the NP-hardness of the problem is straightforward because it is a generalization of the conventional lazy intruder (with well-formed instead of weakly well-formed constraints) and which is already NP-complete [RT01].

For containment in NP, observe that the usual guess-and-check argument is not directly possible, because the smallest solution maybe exponential. For instance, a constraint may have as the only solution: $X_1 = \ll c, c \gg, X_2 = \ll X_1, X_1 \gg, \ldots, X_n = \ll X_{n-1}, X_{n-1} \gg$.

Like [RT01] we use the fact that all these substitutions can however be represented by a polynomial-size DAG. This follows from the termination proof as follows: for a constraint with n variables, we can at most perform n substitutions of variables with more concrete terms throughout the deduction. These substitutions are always with subterms of the initial ϕ. Therefore whatever substitution we can reach, it can be represented by a polynomial size DAG, and we can even effectively enumerate this space because it is bounded by the subterms of ϕ. So we can have a polynomial-time non-deterministic machine guess any of the possible substitutions that can be performed by the calculus. Checking that such a substitution σ is a solution is relatively easy as we sketch it briefly. We do not perform σ to the constraint because this can lead to an exponential blow-up, but we note that we can polynomially check whether for given terms s and t, $\sigma(s) = \sigma(t)$. We first perform the expansion of every intruder knowledge T_i with respect to variables that have not been substituted by σ. Then we analyze each knowledge T_i as far as possible without performing substitutions, i.e., if $t \in T_i$ with $\sigma(t) = \mathit{crypt}(k, m)$, then check whether k can be derived by T_i by only composition (without substitution). Then add m. Again this must all be done without performing σ. E.g., given $\sigma(x) = \mathit{crypt}(x_1, x_2)$ (actually $\sigma(x) = \mathit{crypt}(\sigma(x_1), \sigma(x_2)$—but we assume the DAG representation here), we check that x_1 can be derived, and if so add x_2, but we do not expand to $\sigma(x_1)$ and $\sigma(x_2)$. Similarly, on the construction side we must avoid the expansion. For both analysis and generation we must avoid an exponential number of steps, even though the ground terms have exponential size. The point is that they can only have exponential size by repetition of variables (i.e., they are based on a polynomial-size set of distinct subterms). Thus decomposition cannot run into exponentially many steps; composition can however: consider $\ll X_1, f(X_1) \gg$ and suppose we first compose X_1 and then later $f(X_1)$. This is however easy to avoid: compose pair and f so it remains to compose is (one instance of) X_1. With this it is possible to check the constraints for a given σ in polynomial time. ■

H. Extensions

We now consider a few generalizations of our approach. We did not directly incorporate them into our main exposition, as they unnecessarily complicate the presentation of the core ideas.

First we observe that with weak well-formedness, we still assume the variable origination property, i.e., that we can order constraints so every variable first occurs on the right-hand side of some constraint. In contrast, [ACRT10] consider even constraints where variables are introduced on the left-hand side of a constraint. This means that one intruder receives a message with an undetermined subterm x, so this variable could be instantiated with any ground term. This in particular means that the initial knowledge of the intruders is not necessarily ground. It is possible to handle this kind of constraints with our method, but we need to make some minor modifications. Intuitively, a non-originating variable $x \in T$ means that the respective intruder can "make a wish". Note that the adaption of the semantics, invariant, and proofs is complicating everything without being very insightful. Since we did not see any practical example that would generate constraints with non-originating variables, we decided to keep it out of our main presentation.

Another extension of our approach is the handling of algebraic properties such as the properties of modular exponentiation for Diffie-Hellman based protocols. The problem of the multiple intruder is really *orthogonal* to the problem of handling algebraic equations. In fact we believe that one can "lift" any lazy intruder calculus for well-formed constraints that is correct and terminating for a particular algebraic theory—to one that analogously works on optimistic constraints to be correct and terminating on weakly well-formed constraints. We plan to investigate this claim as part of our future work.

V. Applications

A. Routing Protocols with Multiple Independent Intruders

In this section we show how our procedure can be used to analyze routing protocols for ad-hoc networks. First, We give a modified version of the Secure Routing Protocol SRP [PH02] applied to the Dynamic Source Routing protocol DSR [JMB01]. Then we show that in the extended Dolev-Yao model proposed by Arnaud et al. (where all intruders share their knowledge) there is an attack, but if we consider independent intruder this attack does not exists any more. Finally, we show how we can model the execution of SRP

protocol over our model by a weak-well-formed constraint system, and show that, by solving the obtained constraints, we can get a trace of an attack on SRP protocol.

Let us recall SRP protocol applied to DSR assuming that each node already knows its neighbors and that the source S and the destination D of the route discovery already shared a certain symmetric key K_{SD}. The SRP protocol consists of two phases as follows:

Request phase:

- The source S broadcasts to its neighbors a request message: $\ll req, S, D, Id, [S], H_S \gg$, where Id is a unique identifier, $[S]$ is the initial route list and H_S is the MAC: $hmac_{k_{sd}}(\ll req, S, D, Id \gg)$.
- Each intermediate node that receives the request checks the list, if the list ends with a neighbor's identifier, then it broadcasts the request after appending its identifier to the list, else it drops the message.

Reply phase:

- Once the destination node received a message, it checks that last node in the route is one of its neighbors and verifies the MAC. Then it initiates the reply phase by sending a message containing the discovered route, and a MAC with the key K_{SD}.
- Each intermediate node checks if its identifier appears in the route list between two of its neighbors, if so it forwards the message to the next node in the route list.
- Once the source S receives a message containing a route to D with a MAC matching this route. It checks that the route does not contains loops and that its neighbor in the route is indeed one of its neighbor in the network.

The only difference between MSRP and SRP, is that the destination D initiates the reply phase by sending the discovered route signed with its private key $inv(D)$. So, other nodes have to verify this signature.

We represent the network by an undirected graph $G = (V, E)$ where V is a ground set of nodes made up the network and E is a set of couples that represents the symmetric links between nodes. The statement $(a, b) \in E$ means that a and b are neighbors. All identifiers used in the protocol description that starts with an upper-case letter, e.g. S,D, are variables and we use identifiers that starts with lower-case letters, e.g. s, d, to denote the concrete values used in an execution of the protocol. Note that we use $\ll t_1, t_2, \ldots, t_n \gg$ to represent $\ll t_1, \ll t_2, \ll \cdots, t_n \gg \cdots \gg \gg$ for simplicity.

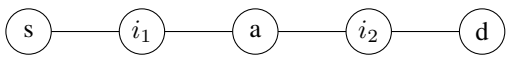

Figure 3. Graph G_0

Consider the network represented in graph G_0 of Figure 3, assume that the node s initiates a route discovery to reach the node d, and that i_1 and i_2 are two intruders. We show an attack exists on MSRP if i_1 and i_2 share their knowledge, and cannot be mounted if they are independent,*i.e.* they do not share there knowledge. First, assume that i_1 and i_2 share their knowledge, then we have the following attack:

Request phase: The source s chooses a unique id and broadcasts $m_1 = \ll req, s, d, id, [s], h_s \gg$, where $h_s = hmac_{k_{sd}}(\ll req, s, d, id \gg)$. The node i_1 receives the route request message m_1. As i_1 and i_2 share their knowledge, then i_1 shears m_1 with i_2. So, i_2 can modify m_1 and obtain a fake message $m'_1 = \ll req, s, d, id, [s, i_1, i_2], h_s \gg$ and send it to d. Since, i_2 is neighbor of d and the MAC h_s computed by k_{sd}, then d accepts m'_1.

Reply phase: The destination d signs the list with $inv(d)$, computes a new MAC $h_d = hmac_{k_{sd}}(\ll req, s, d, id, [s, i_1, i_2, d] \gg)$ and sends $m_2 = \ll rep, s, d, id, sig_{inv(d)}([s, i_1, i_2, d]), h_d \gg$ to i_2, in order to propagate it back to s. When the node i_2 receives the message m_2, i_2 shares it with i_1. In order to complete the attack i_1 just forward m_2 to s. As i_1 is neighbor of s, the MAC of h_d built by d with the correct key k_{sd}, the route $[s, i_1, i_2, d]$ is loop free and signed by $inv(d)$ then s accepts the messages. Thus, i_1 and i_2 can force s and d to believe in the false route $[s, i_1, i_2, d]$ if they share their knowledge.

Now, assume that i_1 and i_2 are independent. We show that they can not force s and d to believe in the false route $[s, i_1, i_2, d]$. The source s initiates the protocol as usual, so i_1 receives the request $m_1 = \ll req, s, d, id, [s], h_s \gg$, where $h_s = hmac_{k_{sd}}(\ll req, s, d, id \gg)$. If i_1 sends m_1 to the node a, then a will drop it since s and a are not neighbors. Another choice for i_1 is to follow the protocol by sending m_1 to a after appending its identifier to the list, then a will accepts it, appends its identifier and sends $\ll req, s, d, id, [s, i_1, a], h_s \gg$ to i_2. Now:

- If i_2 removes a from the list, appends i_2 and sends $m'_1 = \ll req, s, d, id, [s, i_1, i_2], h_s \gg$ to d. Then, d checks the h_s accepts it and initiates the reply phase by sending $m_2 = \ll rep, s, d, id, sig_{inv(d)}([s, i_1, i_2, d]), h_d \gg$ to i_2, where $h_d = hmac_{k_{sd}}(\ll req, s, d, id, [s, i_1, i_2, d] \gg)$. Since, i_2 can not get $inv(d)$, he has no ability to modify the signed list $sig_{inv(d)}([s, i_1, i_2, d])$ by adding a's identifier. So, i_2 cannot pass m_2 through a, since a will drop it as its identifier is not on the list.
- If i_2 follows the protocol and sends $\ll req, s, d, id, l, h_s \gg$ to d, where $l = [s, i_1, a, i_2, d]$. Then, d will reply by $\ll rep, s, d, id, sig_{inv(d)}(l), h'_d \gg$ to i_2, where $h'_d = hmac_{k_{sd}}(\ll req, s, d, id, l \gg)$. Since neither i_1 nor i_2 can modify the signed list $sig_{inv(d)}(l)$, they cannot remove a from the l, and so they cannot force s to believe in the route $l = [s, i_1, i_2, d]$

We show how to model the execution of SRP by a weak well-formed constraint system, and show that, by solving the obtained constraints, we can get a trace of an attack on SRP in the network G_0 of Figure 3 if i_1 and i_2 are independent. Assume that the initial knowledge of i_1 and

i_2 are $T_0^1 = \{s, i_1, a, inv(i_1)\}$ and $T_0^2 = \{a, i_2, d, inv(i_2)\}$ respectively. We assume that a secure neighborhood discovery protocol has been used, consequently, each node can check whether a given node is one of its neighbors. We express these checks thanks to a the following grammar: $P, Q ::= test(\cdot,\cdot), testl(\cdot,\cdot), route(\cdot,\cdot,\cdot), noloop(\cdot), P \wedge Q, P \vee Q, \neg P$, where $test(\cdot,\cdot)$ checks neighborhood property of two nodes, $testl(\cdot,\cdot)$ checks local neighborhood property of a node in a list, $route(\cdot,\cdot,\cdot)$ checks validity of a route between two nodes, $noloop(\cdot)$ verifies that a given list is free from loops.

In Figure 4, we give the semantics of the predicates which models the security properties for analyzing routing protocols, for a given graph $G = (V, E)$, nodes A, B, C, S and D, and list L.

We consider the interaction of the intruders with honest nodes where the exchanged messages are of the form expected by honest nodes, ignoring for a moment the question whether the intruder can generate the respective messages. Using constraints, we model the requirement that the intruder is able to generate each message he sent from his current knowledge. The modeling is as follows:

Request phase: The source s broadcasts $u_1 = \langle req, s, d, id, [s], h_s \rangle$, where $h_s = hmac_{k_{sd}}(\langle req, s, d, id \rangle)$. The first intruder i_1 receives u_1 and adds it to his knowledge, so the knowledge of i_1 becomes $T_1^1 = T_0^1 \cup \{u_1\}$. The node a expects a message of the form $v_1 = \langle req, S_1, D_1, Id_1, L_1 :: N_1, H_1 \rangle$, such that N_1 is neighbor of a. We model this by the constraint: $T_1^1 = T_0^1 \cup \{u_1\} \vdash v_1$ and the formula $P_a = test(a, N_1)$ that reflect, when satisfied, the ability of intruder to build a message of the expected form. The honest node a receives v_1, appends its identifier to $L_1 :: N_1$ and sends $u_2 = \langle req, S_1, D_1, Id_1, L_1 :: N_1 :: a, H_1 \rangle$ to i_2 which add it to his knowledge. Now, i_2 has to build the message $v_2 = \langle req, S_2, D_2, Id_2, L_2 :: N_2, H_2 \rangle$, this results in a new constraint $T_1^1 = T_0^1 \cup \{u_1\} \vdash v_1$ and the formula $P_d = test(N_2, d)$ which grantee that N_2 is neighbor of d.

Reply phase: The destination d computes $h_d = hmac_{k_{sd}}(\langle rep, D_2, S_2, Id_2, L_2 :: N_2 :: d \rangle)$ and sends $u_3 = \langle rep, S_2, D_2, Id_2, L_2 :: N_2 :: d, h_d \rangle$ to i_2. Then i_2 has to build $v_3 = \langle rep, S_2', D_2', Id_2', L_2', H_2' \rangle$ and sends it to a, that result in the constraint $T_3^2 = T_0^2 \cup \{u_2, u_3\} \vdash v_3$. The node a have to check if its identifier appears between two of its neighbors in the list, the formula $P_a' = test(a, L_2')$ reflects this check. So, we assume that a sends $u_4 = v_3$ to i_1. Finally, i_1 has to build $v_4 = \langle rep, S_1', D_1', Id_1', L_1', H_1' \rangle$ and sends it to s, which results on $T_4^1 = T_0^1 \cup \{u_1, u_4\} \vdash v_4$. Since, s has to made neighborhood and loop free checks on the list L_1' the formula $P_s = testl(s, L_1') \wedge noloop(L_1')$ is needed.

The resulting constraint system and logical formula are: $T_1^1 \vdash v_1 \wedge T_2^2 \vdash v_2 \wedge T_3^2 \vdash v_3 \wedge T_4^1 \vdash v_4 \wedge P$, where $P = P_s \wedge P_d \wedge P_a \wedge P_a' \wedge \neg route(s, d, L_1')$.

By setting: S_1, S_1', S_2 and S_2' to s; D_1, D_1', D_2 and D_2' to d; Id_1, Id_1', Id_2, and Id_2' to id; L_1 and L_2 to $[s]$ and $[s, i_1]$ respectively; L_1' and L_2' to $[s, i_1, a, i_2, d]$ and $[s, i_1, i_2, d]$ respectively; N_1 and N_2 to i_1 and i_2 respectively. H_1 and H_2 both to h_s; H_1' and H_2' both to h_d; we get a solution for both the logical formula and the constraint system. This solution is a trace for an attack on SRP in G_0 over our model *i.e,* i_1 and i_2 are independent, where s and d can be convinced that $[s, i_1, i_2, d]$ is a valid route between them.

B. The Mobile Intruder

In this section we consider the problem of a platform that executes some code from an intruder, e.g., a web-server running a script from a potentially malicious website, a mobile phone running a downloaded application, or a virtual machine that is hosting application from potentially dishonest customers. We give an extension of our work in [MNN13] which uses the mobile ambient calculus of Cardelli and Gordon [CG00] to model the platforms and the code they are hosting, and applies the constraint-based lazy intruder technique to efficiently analyze security problems in this calculus.

The ambient calculus is a process calculus with the usual constructs like the parallel composition $P \mid Q$ of two processes P and Q. An *ambient* is a process with a boundary around it, written $n[P]$ where P is a process and n is the name of the ambient. For instance $P_1 \mid p[P_2 \mid v_1[P_3] \mid v_2[P_4]]$ could model a platform p that is running a process P_2 as well as two virtual machines v_1 and v_2 that run processes P_3 and P_4, respectively; outside the platform runs another process P_1. There are three kinds of *capabilities*: $in\ n$, $out\ n$, and $open\ n$ to enter, exit, or open an ambient n, respectively. For instance the process $n[in\ m.P] \mid m[Q]$ can reduce to $m[n[P] \mid Q]$. Observe here that only an entire ambient (like $n[\cdot]$) can move. Cardelli and Gorden also consider an extension of the basic ambient calculus with the communication of capabilities. A process can only communicate when they are in parallel with each other, namely: $\langle M \rangle \mid (x).P \Rightarrow \sigma(P)$ where σ substitutes x with M.

Our paper [MNN13] considers the execution of one or more malicious processes in an environment of honest participants. The malicious processes are constructed by an intruder from his initial knowledge. The initial knowledge consists of a set of (ground) ambient names and capabilities. The intruder can construct processes from his knowledge by using the constructors of the ambient calculus, just like he can construct messages in protocol verification. For instance knowing n and $in\ m$, he can construct the process $n[in\ m.(x).out\ x]$. Note that the intruder may use arbitrary variable symbols in a process, but the process must be *closed*, *i.e.* every variable must be bound by some surrounding input operation.

We are interested in the following problem. Let us call a context $C[\cdot]$ "a process with a whole", *i.e.* with a subterm

$\mathcal{I}, G \models test(A, B)$ iff $(\mathcal{I}(A), \mathcal{I}(B)) \in E$ or $(\mathcal{I}(B), \mathcal{I}(A)) \in E$
$\mathcal{I}, G \models testl(C, L)$ iff $\mathcal{I}(C)$ appears on $\mathcal{I}(L)$ between two of its neighbors.
$\mathcal{I}, G \models noloop(L)$ iff $\mathcal{I}(L) = a_1 :: \cdots :: a_n$ such that, for every $1 \leq i, j \leq n$, if $i \neq j$ then $a_i \neq a_j$.
$\mathcal{I}, G \models route(S, D, L)$ iff $\mathcal{I}(L) = a_1 :: \cdots :: a_n$ such that $\mathcal{I}(S) = a_1, \mathcal{I}(D) = a_n$, for every $1 \leq i < n$, $(a_i, a_{i+1}) \in E$, and for every $1 \leq i, j \leq n$, if $i \neq j$ then $a_i \neq a_j$.
$\mathcal{I}, G \models P \wedge Q$ iff $\mathcal{I}, G \models P$ and $\mathcal{I}, G \models Q$
$\mathcal{I}, G \models P \vee Q$ iff $\mathcal{I}, G \models P$ or $\mathcal{I}, G \models Q$
$\mathcal{I}, G \models \neg P$ iff $\mathcal{I}, G \not\models P$

Figure 4. Semantics of the interpretations for formulas

where we can insert any term P of type process and to obtain the process $C[P]$. Given the following:

- an honest platform as a context $C[\cdot]$
- an initial intruder knowledge T_0,
- and an attack predicate $attack(P)$ that formalizes that in P an attack has occurred

can the intruder construct any process P_0 from T_0, written $T_0 \vdash P_0$ such that $C[P_0] \rightarrow^* P$ and $attack(P)$, *i.e.* can the platform upon executing any intruder process ever reach an attack state?

Note that even though we consider here only one initial intruder process, this process may be a parallel composition of several sub-processes and they may move to different locations. For instance in above example of the host p with two virtual machines, we may have that the intruder process P_0 is initially outside the host p. Then a sub-process of P_0 may move into the virtual machine v_1 and learn there a secret s_1, and another sub-process of P_0 may similarly move into v_2 and there learn s_2. The desired security property may for instance be that neither process can ever communicate s_1 or s_2 to a process outside the host p and that no intruder process ever sees both secrets s_1 and s_2.

Our approach is to use the lazy intruder technique to answer this kind of questions: instead of directly exploring the infinite space of processes P_0 that the intruder can construct from his initial knowledge T_0, we work with a placeholder $\boxed{T}$ that represents *any* process that the intruder can construct from knowledge T and instantiate it during the search. For instance, from the state $n[\boxed{T} \mid R] \mid m[Q]$ we can reach $m[n[\boxed{T} \mid R] \mid Q]$ if $T \vdash in\ m$. The reason is that if the intruder has the capability $in\ m$, he can construct the process $P = in\ m.P_0$ from T for any process $P_0 \dashv T$. Thus, from $n[P \mid R] \mid m[Q]$ we can reach $m[n[P_0 \mid R] \mid Q]$. The $\boxed{T}$ notation here allows us to work without the variables P and P_0 and focus on constraints like $T \vdash in\ m$ that deal only with capabilities (not processes).

When we look at communication, we get a very natural effect with the lazy intruder: if the intruder code runs in parallel with an honest process who wants to communicate a capability M, then this capability is simply added to the intruder knowledge: $\langle M \rangle \mid \boxed{T} \rightarrow \boxed{T \cup \{M\}}$. This rule is sound because for every process P that the intruder can construct from knowledge $T \cup \{M\}$ he can analogously build a process $(x).P'$ from knowledge T that contains the variable x at every position where P uses the capability M. Thus, replacing in P' every occurrence of x with M gives P.

Vice versa, if there is an honest process who would like to receive a message, we can use the laziness of the intruder to postpone the decision which concrete message should be sent. The transition rule is therefore simply $(x).P \mid \boxed{T} \rightarrow P \mid \boxed{T}$ where $T \vdash x$. Note that P is not a closed process anymore, but has a free variable x (unless input x is never used in P). We add however the intruder constraint that x must be some capability that can be constructed from T.

Finally, if two intruder processes meet, they can *pool* their knowledge: $\boxed{T} \mid \boxed{T'} \rightarrow \boxed{T \cup T'}$. This is sound because the intruder can design the first process so that it successively sends every capability of $T \setminus T'$ and the second process to receives them.

From the lazy intruder approach we thus get a symbolic state transition system where each state consists of a process and a conjunction of $T \vdash M$ constraints where T is a set of capabilities and M is a capability (where both T and M may contain variables). Every state in this symbolic transition system has finitely many successor states and if we do not have unbounded replication of honest processes (*i.e.* , $!P$) we can safely limit ourselves to exploring the transition system to a finite depth [MNN13]. *Safely* here means that if an attack state is reachable then one exists before the depth bound. Thus what remains is to check the satisfiability of the intruder constraints.

Here we have again the multiple-intruder problem, even if we start with only one intruder process with ground initial knowledge T_0. This is because the intruder process can branch into several ones that move and learn independently in the contact with honest agents. Consider for instance the process $\boxed{T} \mid n[\boxed{T'} \mid (x).m[out\ n.\langle x \rangle]]$. Here, inside the ambient n we have an intruder running parallel with an agent who wants to receive some x and then move outside the ambient n and output x. Lazily we thus get first $\boxed{T} \mid n[\boxed{T'} \mid m[out\ n.\langle x \rangle]]$ with the constraint $T' \vdash x$. Then we get with another transition to $\boxed{T} \mid m[\langle x \rangle] \mid n[\boxed{T'}]$. Suppose $open\ m \in T$, then we can reach $\boxed{T} \mid \langle x \rangle \mid n[\boxed{T'}]$ and thus $\boxed{T \cup \{x\}} \mid n[\boxed{T'}]$. Thus, the process $\boxed{T \cup \{x\}}$ has learned some value x that was generated from knowledge T', and T' is not necessarily a subset of T.

As in the case of wireless ad-hoc networks, we thus

produce here intruder constraints that are no longer well-formed in the classical sense. In the example for instance, any further constraint caused by the process $\boxed{T \cup \{x\}}$ will be of the form $T \cup \{x\} \vdash M$ for some capability M, and here we have a variable x in the knowledge side that originates in a constraint $T' \vdash x$; the knowledge T' is not necessarily a subset of T. However, as in the wireless scenario, we do have the weak well-formedness: we have an order on the constraints (the order in which they have been created) according to which every variable first occurs on the right-hand side of a $\vdash$, *i.e.* every variable is still related to the choice of *some* intruder rather than being an arbitrary value.

In fact, for the ambient calculus as considered in [MNN13], this weaker form causes no trouble because the constraints only deal with capabilities and there are no "analysis" rules on that, *i.e.* , from a term that describes a capability we can never learn a subterm. This is in contrast to messages in cryptographic protocols, where an intruder can for instance learn the clear-text of an encrypted message in his knowledge, if he also knows the corresponding key for decryption. In fact, analysis is in a way the tricky part when we have constraints that are not well-formed in the classical sense.

Extending Ambients with Arbitrary Messages: An interesting extension of ambients is to allow the communication of cryptographic messages between processes, so that we can easily model communication protocols occurring between the processes. The reason is that within a platform (e.g. the network of a company, a virtual machine, or a browser) the communication between processes may also need properties like authentication and confidentially because not all processes are necessarily honest.

In the ambient calculus, the term capability is defined as the least closure of (ambient) names and variables under the constructors in, out, and $open$. Communication is defined as $\langle M \rangle$ and $(x).P$, and actions are of the form $M.P$ where M is capability x is a variable and P is a process. We now propose the following conservative extension: capabilities may be arbitrary terms from the message algebra $\mathcal{T}(\mathcal{F}, V)$, and where in, out, and $open$ are simply unary function symbols of $\mathcal{F}$ that are intrudable. Of course all occurring variables in a process should be bounded in the end as it is standard, *i.e.* , we do not consider processes with free variables (except during evaluations with the lazy intruder where still all variables are "bounded" by some constraint). We also want to allow $(M).P$ on an input action where M may contain variables. This is for pattern matching and means that the process is willing to receive any ground message that is an instance of M, thereby binding the variables in M and continue with an appropriate instance of P. For instance $(acrypt_b(\ll x_{NA}, x_A \gg)).\langle acrypt_{x_A}(\ll x_{NA}, nb \gg)\rangle.(acrypt_b(nb)).n[in\ m.\langle x_{NA}, nb\rangle]$ would be a process that first behaves like the receiver role in the famous Needham-Schroeder protocol and when that is successfully finished enters the ambient m and outputs the agreed pair of nonces in that protected environment.

Formally, the semantics of the ambient calculus is defined operationally as a reduction relation on closed processes; we change it by only generalizing the communication rule for the described pattern matching: $\langle M \rangle \mid (M').P \to \sigma(P)$ if $\sigma \in mgu(M, M')$. Note that this is on closed processes, so M is a ground term, thus σ substitutes all variables of M' with ground terms. Thus since $(M').P$ is closed, also $\sigma(P)$ is.

One may wonder whether this needs other extensions, since a process can read a variable and then perform that as an action, so for instance $\langle scrypt_k(m)\rangle \mid (x).x.0 \to scrypt_k(m).0$. In fact, we do not forbid that (just like the original calculus does not forbid ill-formed capabilities like $in\ in\ m$) and just observe that this process cannot go any further because no rule matches it.

For the lazy intruder approach of checking mobile ambients we do not get many changes either from this extension—in particular the communication rules stay exactly the same except the intruder output. Here, due to the pattern matching, we need to generalize the rule as follows: $(M).P \mid \boxed{T} \to P \mid \boxed{T}$ where $T \vdash M$. *i.e.* , rather than a single variable x the intruder here has to construct whatever term M is. As a result we get a conjunction of weak-well-formed constraints on messages. Note that the weak variable origination property (*i.e.* each variable first occurs on the right-hand side of a constraint) still holds because every free variable of the process P in the above rule must occur in M.

VI. Conclusion

We have shown how to decide the satisfiability of *weak-well-formed* constraint systems. Our procedure reduces a given *weak-well-formed* constraint system into a set of simpler ones, until we either reach a simple (or solved) form or we cannot do further reductions. The correctness proof shows that set of all reachable constraint systems we can reach by these reductions are together equivalent to the original constraint system. Moreover, when a satisfiable constraint system is non-simple, there is an applicable reduction rule. Thus, if we reach an irreducible, non-simple constraint system, it must be unsatisfiable. The termination proof ensures that the reduction of constraint systems does not admit infinitely long non-redundant deductions. Moreover we can easily see that the problem is still in NP (NP-hardness follows because it is a generalization of an NP-complete problem). We use this result about multiple independent intruders to analyze routing protocols in wireless networks where each intruder is a single node of the network who cannot directly communicate with other intruders outside his proximity. We further use it to extend the result of [MNN13] to communication of arbitrary cryptographic messages; this is helpful for analyzing platforms like web-browsers, mobile phones, and virtualized infrastructures that host potentially malicious code.

REFERENCES

[ACD10] Mathilde Arnaud, Véronique Cortier, and Stéphanie Delaune. Modeling and verifying ad hoc routing protocols. In *CSF*, pages 59–74. IEEE Computer Society, 2010.

[ACRT10] Tigran Avanesov, Yannick Chevalier, Michaël Rusinowitch, and Mathieu Turuani. Satisfiability of general intruder constraints with a set constructor. In *CRiSIS*, pages 1–8. IEEE, 2010.

[ACRT12] Tigran Avanesov, Yannick Chevalier, Michaël Rusinowitch, and Mathieu Turuani. Intruder deducibility constraints with negation. decidability and application to secured service compositions. *CoRR*, abs/1207.4871, 2012.

[BMV05] David Basin, Sebastian Mödersheim, and Luca Viganò. OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208, 2005.

[CG00] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theor. Comput. Sci.*, 240(1):177–213, 2000.

[CKRT03] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents. In *FST TCS'03*, LNCS 2914, pages 124–135, 2003.

[CLCZ10] Hubert Comon-Lundh, Véronique Cortier, and Eugen Zalinescu. Deciding security properties for cryptographic protocols. application to key cycles. *ACM Trans. Comput. Log.*, 11(2), 2010.

[CLS03] Hubert Comon-Lundh and Vitaly Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *LICS*, pages 271–. IEEE Computer Society, 2003.

[DLLT08] Stéphanie Delaune, Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Symbolic protocol analysis for monoidal equational theories. *Inf. Comput.*, 206(2-4):312–351, 2008.

[HPJ06] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):370–380, 2006.

[Hui99] Antti Huima. Efficient infinite-state analysis of security protocols. In *Proc. FLOC'99 Workshop on Formal Methods and Security Protocols*, 1999.

[JMB01] David B. Johnson, David A. Maltz, and Josh Broch. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In *In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5*, pages 139–172. Addison-Wesley, 2001.

[LPM+05] Loukas Lazos, Radha Poovendran, Catherine Meadows, Paul F. Syverson, and LiWu Chang. Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach. In *WCNC*, pages 1193–1199. IEEE, 2005.

[MNN13] Sebastian Mödersheim, Flemming Nielson, and Hanne Riis Nielson. Lazy mobile intruders. In *POST 2013*, 2013. To appear, available as Tech Report IMM-TR-2012-13 at imm.dtu.dk/~samo.

[MS01] Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In Michael K. Reiter and Pierangela Samarati, editors, *ACM Conference on Computer and Communications Security*, pages 166–175. ACM, 2001.

[MS03] Jonathan K. Millen and Vitaly Shmatikov. Symbolic protocol analysis with products and diffie-hellman exponentiation. In *CSFW*, pages 47–61. IEEE Computer Society, 2003.

[PH02] Panagiotis Papadimitratos and Zygmunt Haas. Secure routing for mobile ad hoc networks. *MOBILE COMPUTING AND COMMUNICATIONS REVIEW*, 1(2):27–31, 2002.

[RT01] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with finite number of sessions is np-complete. In *Theoretical Computer Science*, pages 174–190, 2001.

[RT03] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with a finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 1-3(299):451–475, 2003.

APPENDIX

Lemma IV.2 (Closure)**.** *For every ϕ, the optimistic constraint $\phi[]\phi$ satisfies the invariant. Application of the function $expand$ and application of a substitution σ to $\phi[V]\phi'$ preserves the invariant and the weak well-formedness.*

Proof: The invariant holds for $\phi[]\phi$ because $\phi \models \phi$.

Preservation of the invariant under $expand$: Let $\phi[V]\phi'$ be a constraint that satisfies the invariant, and let $\phi' = \phi'_0 \wedge T \vdash t$ such that there exists a variable $x \in T$. After applying $expand$, we thus have $\phi[V \cup \{x\}]\phi''$ for $\phi'' = \phi'_0 \wedge T' \vdash t$ where $T' = T \setminus \{x\} \cup know_{\phi'}(x)$. We have to show $\phi'' \wedge R_{V\cup\{x\}} \models \phi$:

$$\phi'' \wedge R_{V\cup\{x\}} \models \phi'_0 \wedge T' \vdash t \wedge \{x\} \vdash know_{\phi'}(x) \wedge R_V \models \phi'_0 \wedge T \vdash t \wedge R_V \models \phi' \wedge R_V \models \phi$$

The last implication uses the fact that invariant already holds for $\phi[V]\phi'$.

Preservation of weak well-formedness under $expand$: All variables in $know_\phi(x)$ must originate even earlier in the constraint according to the order of the weak-well-formedness, so we cannot destroy the origination property.

Preservation of the invariant under substitutions: In the first case, the invariant is preserved since $\sigma(\phi) \wedge eq(\sigma) \models \sigma(\phi) \wedge eq(\sigma)$ and expand preserves the invariant. For the second case, first observe that from $\phi \models \psi$ follows $\sigma(\phi) \models \sigma(\psi)$ for any ϕ,ψ, and σ.[2] Since the invariant holds for $\phi[V]\phi'$ we have $\phi' \wedge R_V \models \phi$. Since $dom(\sigma) \cap V = \emptyset$, we conclude $\sigma(\phi') \wedge R_V \wedge eq(\sigma) \models \sigma(\phi) \wedge eq(\sigma)$.

Preservation of weak well-formedness under substitutions: Instantiation of variables cannot destroy the variable origination property. About the occurrence of variables in equations: for every $x = t$ in $eq(\sigma)$, x cannot occur on the left-hand side of an equation in ϕ or ϕ' and does not occur in $\sigma(\phi)$ and $\sigma(\phi')$. ∎

Lemma IV.3 (Soundness)**.** *All the constraint rules are sound, i.e., for every concrete reduction $\frac{\chi'}{\chi}$ with our calculus holds $\chi' \models \chi$.*

Proof: Note that by the semantics of optimistic constraint stores $\phi[V]\phi'$ only ϕ is relevant. Therefore, both generate and analysis rules are straightforward. For the unify rule, we have the case $\sigma(\phi[V]\ldots) \equiv \sigma(\phi) \wedge eq(\sigma) \models \phi$, which is immediate. ∎

[2]For first-order logic this is straightforward, using the deduction theorem; however since our constraints are interpreted in a way that involves the least transitive closure of the deduction relation, we cannot directly use that argument. Let $\phi \models \psi$ and let $\mathcal{I}_\sigma$ be any model of $\sigma(\phi)$. Then construct another interpretation $\mathcal{I}$ as follows:

$$\mathcal{I}(x) = \begin{cases} \mathcal{I}_\sigma(\sigma(x)) & \text{if } x \in dom(\sigma) \\ \mathcal{I}_\sigma(x) & \text{otherwise} \end{cases}$$

Then $\mathcal{I} \models \phi$ and thus $\mathcal{I} \models \psi$. Since $\sigma(\psi)$ does not contain any variables of $dom(\sigma)$ and $\mathcal{I}_\sigma$ is identical with $\mathcal{I}$ on all other variables, also $\mathcal{I}_\sigma \models \sigma(\psi)$.

Lemma IV.4 (Completeness)**.** *Given a weakly well-formed conventional constraint ϕ. Starting with the optimistic constraint* $\mathsf{expand}(\phi[V]\phi)$*, all optimistic constraints that we can reach with the reduction rules are either simple, unsatisfiable, or admit the application of another reduction rule.*

Proof: First observe that by the soundness, no reduction rule can introduce new solutions. It therefore suffices to fix one model $\mathcal{I}$ of the original ϕ (if ϕ is unsatisfiable, the lemma trivially holds by soundness) and show that we can always find a reduction that supports $\mathcal{I}$ unless we have run into a simple constraint.

Since $\mathcal{I} \models \phi$, for every $T \vdash t$ we know $\mathcal{I}(T) \vdash \mathcal{I}(t)$. Therefore there exists an intruder derivation tree, i.e., a tree of ground terms, where the root is $\mathcal{I}(t)$, the leaves are elements of $\mathcal{I}(T)$ and every inner node can be derived from its children by one step of the Dolev-Yao model of Fig. 2. Let us therefore label in $\mathsf{expand}(\phi[]\phi)$ every term $t \dashv T$ with a derivation tree for $\mathcal{I}(t)$ with leaves in $\mathcal{I}(T)$. (This is also possible under expand, $T\setminus\{x\}\cup know_\phi(x)$ allows at least as much to derive as T when $x \in T$.) We show throughout the reduction we can preserve this labeling for some constraint and consider an arbitrary $\phi[V]\phi'$ that we have reached and that is accordingly labeled.

At any state $\phi[V]\phi'$, we focus on the first conjunct $T \vdash t$ of ϕ' (in the order of the weak well-formedness) that is not simple, i.e., $t \notin \mathcal{V}$, and the derivation tree of $\mathcal{I}(t)$:

- If the derivation tree is a leaf, then $\mathcal{I}(t) \in \mathcal{I}(T)$. Since $t \notin \mathcal{V}$ and $T \cap \mathcal{V} = \emptyset$ (due to the application of $expand$), the unify rule is applicable with respect to t and some term $s \in T$ such that $\mathcal{I}(t) = \mathcal{I}(s)$. The most general unifier σ of s and t must therefore be supported by $\mathcal{I}$, i.e., $\mathcal{I} \models eq(\sigma)$. Thus, $\mathcal{I}$ is still a model of the new optimistic constraint store we obtain by the unify rule. Moreover, we obtain a correct labeling of the $T \vdash t$ conjuncts, if we just keep all labels as they are: since $\mathcal{I} \models eq(\sigma)$, it is correct if $\sigma(T) \vdash \sigma(t)$ is labeled by a derivation tree for $\mathcal{I}(t)$ with leaves in $\mathcal{I}(T)$.
- If it is a composition step, then since $t \notin \mathcal{V}$, we have $t = f(t_1, \ldots, t_n)$, and f must be intrudable, because the composition step works on the ground level and $\mathcal{I}(t) = f(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n))$. Therefore the generate rule is applicable, and the $T \vdash t_i$ constraints in the new constraint system can be correctly labeled with the respective subtrees for $\mathcal{I}(t_i)$.
- If it is an analysis step, note that the analysis rule may not be directly applicable, if the term being analyzed is itself a subtree that contains further analysis (or composition) steps, so we need to consider the derivation tree of the term being analyzed.
  Here we can exclude the case that a term being analyzed is composed before, e.g., an intruder first encrypts a message and then decrypts it again (in such cases we can easily simplify the proof tree as expected). However,

that does not mean that the proof tree with an analysis node at the root cannot have composition steps: e.g. for the symmetric encryption $scrypt_{h(k_1,k_2)}(m)$, if the intruder knows both k_1 and k_2 and h is intrudable, he is able to first compose the key and then analyze the encrypted message to obtain m. What we can exclude is only that the derivation tree for a term t contains as a proper subtree again a derivation tree for t. From that we derive that for any analysis step in the derivation tree the analyzed term itself is either a leaf (i.e., in $\mathcal{I}(T)$) or the result of an analysis step. Since the derivation tree cannot have infinitely long paths, following the analyzed terms eventually leads us to a leaf, i.e., a term in $\mathcal{I}(T)$ that can be analyzed in $\mathcal{I}$.
Recall $T \cap \mathcal{V} = \emptyset$ (because of the application of expand). Therefore there must be a term $t_0 \in T$ for which analysis can be applied, and in fact it can be applied to every other conjunct $T' \vdash t'$ with $T' \supseteq T$. In the constraint system we obtain after the analysis step, we may have a new constraint for the key derivation, if the analysis step was a decryption, e.g., in the asymmetric $T \vdash inv(k)$. In the derivation tree of the analyzed term $\mathcal{I}(t_0)$, there is thus a subtree for the key, in the example for $\mathcal{I}(inv(k))$. We can thus label the key derivation with an appropriate derivation tree (with leaves in $\mathcal{I}(T)$). In all constraints that are updated by the $m \gg T$, i.e., where we have derivation trees with leaves in $\mathcal{I}(T')$ for some $T' \supseteq T$, we can replace all analysis steps for $\mathcal{I}(t_0)$ simply with a leaf since the respective subterm of t_0 is now added to T' by the update.

■

Lemma IV.5 (Termination)**.** *Given the optimistic constraint store* $\mathsf{expand}(\phi[]\phi)$ *for some weakly well-formed conventional constraint* ϕ*. Then the reduction rules do not admit an infinitely long sequence of non-redundant reduction steps.*

Proof: For optimistic constraints $\phi[V]\phi'$, we define a termination order on pairs (k, w) of positive integers as $(k, w) > (k', w')$ iff $k > k'$ or ($k = k'$ and $w > w'$). This order is well-founded. The first component k is the number of variables that occurs in ϕ, ϕ' and V without taking in the account the equality constraints ($x = t$), and w is the maximum number of reduction steps that can be done without substituting any variable. This value w can be computed on ϕ' as follows:

$$w(\phi \wedge \psi) = w(\phi) + w(\psi) \tag{1}$$
$$w(T \vdash t) = w_1(T) + w_2(t) \tag{2}$$
$$w_1(\{t_1, \ldots, t_n\}) = w_2(t_1) + \ldots + w_2(t_n) \tag{3}$$
$$w_2(f(t_1, \ldots, t_n)) = 1 + w_2(t_1) + \ldots + w_2(t_n) \tag{4}$$
$$w_2(x) = 0 \tag{5}$$

It is immediate that all non-redundant reduction steps that do not substitute any variable (including the Unify rule when applied to identical terms s and t) reduce the weight $w(\phi')$. In a step that does unify a variable, then the number of the variables in the resulting constraint is lower (because in the free algebra $mgu(\cdot)$ never introduces new variables); thus this reduces the first component k of the order—while the second component may reset to a higher value when a variable of V is instantiated. Thus every reduction step decreases the weight of the constraint and it is positively defined, so there cannot be an infinite derivation using non-redundant reductions. ■