ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Technical Report No. 405

Constraint Differentiation: A New Reduction Technique for Constraint-Based Analysis of Security Protocols*

David Basin   Sebastian Mödersheim   Luca Viganò

Information Security Group, ETH Zentrum, CH-8092 Zurich, Switzerland

www.infsec.ethz.ch/~{basin,moedersheim,vigano}

{basin,moedersheim,vigano}@inf.ethz.ch

May 16, 2003

Abstract

We introduce constraint differentiation, a new technique for reducing search when model-checking security protocols. Our technique is based on eliminating certain kinds of redundancies that arise in the search space when using symbolic exploration methods, in particular methods that employ constraints to represent and manipulate possible messages from an active intruder. Formally, we prove that constraint differentiation terminates and is correct and complete, in that it preserves the set of reachable states so that all state-based properties holding before reduction (such as the existence of an attack) hold after reduction. Practically, we have integrated this technique into OFMC, a state-of-the-art model-checker, and demonstrated its effectiveness by extensive experimentation. Our results show that constraint differentiation substantially reduces search and considerably improves the performance of OFMC, enabling its application to a wider class of problems.

*This work was supported by the projects IST-2000-26410, "AVISS: Automated Verification of Infinite State Systems", and IST-2001-39252, "AVISPA: Automated Validation of Internet Security Protocols and Applications".

1 Introduction

Context. A wide variety of model-checking approaches have recently been developed to analyze security protocols, e.g. [5, 9, 12, 14, 23, 24, 28]. The challenge faced when building such a checker is to handle two kinds of state-explosion. The first kind is caused by the standard Dolev-Yao [13] model that defines an infinite set of messages that the intruder can generate.[1] The second kind stems from the large number of possible interleavings resulting from parallel executions of a protocol by the honest agents and the intruder.

A number of constraint-based approaches have been proposed to tackle the first problem, which employ symbolic representation techniques [17, 6, 1, 23, 15, 8, 9, 12, 7, 5]. Although there are differences between these approaches, they all share in common a symbolic representation of the state space, i.e. sets of (ground) states are represented by terms with variables and constraints on the variables. These constraints describe what terms an intruder must generate from a given set of known messages according to the Dolev-Yao model. Moreover, all these approaches use similar reduction rules to manipulate the constraints; in particular, constraint reduction is demand-driven ("lazy") in the sense that one checks only if a solution for the constraints exists, rather than exploring all solutions. Therefore, we will refer to the technique underlying these constraint-based approaches as the *lazy intruder technique*.

There are also structural similarities in the different realizations of the lazy intruder as we can generally distinguish two *layers of search*: the first layer is a search in the space of constraints to find all (symbolic) solutions for a given set of constraints according to the Dolev-Yao model; the second layer builds on the first one to search in the symbolic search space to determine if an *attack state* is reachable. The first kind of search is performed in a backward fashion ("Can the terms to be generated be reduced to terms the intruder knows?"), while the second kind is performed in a forward fashion ("What states are reachable from the initial state?"). While the different approaches implement the first layer in similar ways, there are differences in the formalism employed to represent the second layer, e.g. [5, 9] use *multiset rewriting* to represent the symbolic state space and its transition relation, [12, 23] use *strand spaces*, and [1, 7] use *process calculi*.

The lazy intruder drastically reduces the search tree generated during protocol analysis, providing an effective solution to the first problem mentioned above, the prolific Dolev-Yao intruder. However, the second problem is still open, namely reducing the state-explosion that results from the exponential number of interleavings induced by parallel protocol executions.

In standard model-checking approaches for concurrent systems, the state-explosion problem is often handled using *partial-order reduction* (*POR*), a technique that reduces the number of interleavings which need to be considered by exploiting independencies between the possible transitions [26]. One might expect that the direct combination of the lazy intruder with partial-order reduction as part of the second layer of the search (i.e. while searching the symbolic state space) would allow us to simultaneously "solve" both state-explosion problems. However, as we will explain below, this combination is not effective: the different transitions of the lazy intruder rarely lead to the same successor state, and therefore there is practically no independence of transitions that can be exploited by POR.

Contribution. We show here how to effectively integrate the lazy intruder and ideas from POR by using independence information from the second layer, i.e. the symbolic transition system, when

[1] Using restrictions on the form of messages the intruder may generate (e.g. by allowing only type-correct messages), it is possible to obtain a finite model of the intruder. However, the branching factor of the search tree induced by such a restricted Dolev-Yao intruder is typically still enormous.

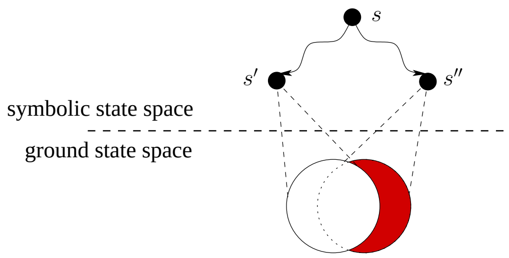

Figure 1: The intuition behind constraint differentiation.

searching the first layer, i.e. the constraint reduction. We call the resulting technique *constraint differentiation*.

Figure 1 illustrates the intuition behind constraint differentiation. Consider two symbolic states s' and s'' that can be reached from some state s by different interleavings of the same actions, e.g. message exchanges. In practice, there is often a substantial overlap between the sets of ground states represented by s' and s''. Constraint differentiation exploits these overlaps by restricting the constraints of s'' to those ground states that are not already covered by s', i.e. the shaded part in Figure 1. (Symmetrically, we could restrict s' instead of s''.)

Our work has both theoretical and practical relevance. Theoretically, our contribution is the formalization of constraint differentiation as a search reduction technique integrating the lazy intruder with ideas from POR. We formally prove that constraint differentiation terminates and is correct and complete, in the sense that it preserves the set of reachable states so that all state-based properties holding before reduction (such as the existence of an attack) hold after reduction. It follows that constraint differentiation neither excludes attacks nor introduces new ones.

Constraint differentiation is independent of the technical and conceptual details of the various lazy intruder approaches and underlying protocol models, and can thus be adopted in other approaches. In particular, even though for concreteness we refer to our own variant of the lazy intruder technique [5], it is not difficult to carry over constraint differentiation to other approaches, as the different formalisms for constraint reduction in the first layer of search are all similar. For the second layer of search, we abstract away from particular base formalisms and work only with the requirement that the state space is represented symbolically, in the sense that it can be expressed using terms with variables, and attacks are formulated as reachability problems. To this end, we introduce the notion of a *symbolic transition system*, which provides an interface to the different formalisms.

Our theoretical results have immediate practical applications. As a concrete example, we have taken the on-the-fly model-checker OFMC [5], a state-of-the-art tool for security protocol analysis based on the lazy intruder, and we have integrated our proposed constraint differentiation into this tool. This integration generally reduces the search time by a factor 2 to several orders of magnitude. More importantly, this improvement extends the scope of the OFMC tool so that it not only scales well to the falsification of industrial-strength protocols, but it can also be applied as an effective verification tool since our reductions make it feasible to exhaustively search the (symbolic) state space resulting from several parallel executions of the protocol being analyzed. We have carried out extensive experiments to validate our approach. As a relevant example, we report on our results for the protocol-suite IKE.

Organization. We proceed as follows. In §2 we formalize the lazy intruder and symbolic transition systems. In §3 we formalize the integration of constraint differentiation into this symbolic protocol model. We present experimental results in §4 and draw conclusions and discuss future work in §5. Due to lack of space, examples and proofs have been shortened or moved to the appendix.

2 Constraint-Based Protocol Models

We introduce the basic constraint-based model underlying our approach in a bottom-up fashion, starting with the first layer of the search, the constraints and their reduction. Then, for the second layer, we introduce symbolic transition systems by abstracting away the details of the formalism used to represent the symbolic state space.

We will proceed as abstractly as possible, without limiting the design choices of the approach, and will only commit to particular choices (based on [5]) to make the exposition concrete.

2.1 The Lazy Intruder and Constraint Reduction

As is standard, messages in our model are elements of an algebra $T_\Sigma(\mathcal{V})$, the set of *terms* built from a signature Σ and a set of variables $\mathcal{V}$. The signature Σ contains the operators for building messages, and for brevity we restrict here our presentation to *pairing*, denoted by $\langle m_1{,}m_2\rangle$, and *symmetric encryption*, denoted by $\{\!|m_1|\!\}_{m_2}$; note that arbitrary messages can be used as keys, allowing us to analyze protocols with non-atomic keys. It is straightforward to extend the methods presented in this paper to other operators like asymmetric encryption, hashing, and key-tables. Note that we do not associate any type information with messages in order to allow the detection of type-flaw attacks. Also, like most other approaches, we employ the *free algebra assumption* and assume that syntactically different terms represent different messages.

The notions of *atomic* and *composed message* are defined in the usual way and so are the notions related to substitution and unification, such as *ground term*, *(ground) substitution*, *unifier*, and *most general unifier (mgu)*; see, e.g., [3]. We denote the application of a substitution σ to a term t by writing $t\sigma$ and denote the composition of substitutions σ_1 and σ_2 by writing $\sigma_1\sigma_2$. As we only consider substitutions with finite domains, we represent a substitution σ with $domain(\sigma) = \{v_1, \ldots, v_n\}$ by $[v_1 \mapsto v_1\sigma, \ldots, v_n \mapsto v_n\sigma]$. The *identity substitution* id is the substitution with $domain(\mathrm{id}) = \emptyset$. We say that two substitutions σ_1 and σ_2 are *compatible*, written $\sigma_1 \approx \sigma_2$, if $v\sigma_1 = v\sigma_2$ for every $v \in domain(\sigma_1) \cap domain(\sigma_2)$. For two sets of ground substitutions Σ_1 and Σ_2, we define their *intersection modulo the different domains* as $\Sigma_1 \sqcap \Sigma_2 = \{\sigma_1\sigma_2 \mid \sigma_1 \in \Sigma_1 \ \wedge \ \sigma_2 \in \Sigma_2 \ \wedge \ \sigma_1 \approx \sigma_2\}$. Since the composition of compatible ground substitutions is associative and commutative, so is the $\sqcap$ operator.

We follow Dolev and Yao [13] and consider the standard asynchronous intruder model where the intruder controls the network but cannot break cryptography. In particular, the intruder can intercept messages and analyze them if he possesses the corresponding keys for decryption, and he can generate messages from his knowledge and send them under any agent name.

Definition 1. *For a set M of messages, let $\mathcal{DY}(M)$ (for Dolev-Yao) be the smallest set closed under the following* generation (G) *and* analysis (A) rules*:*

$$\frac{m \in M}{m \in \mathcal{DY}(M)}\ G_{\text{axiom}}\,, \quad \frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{\langle m_1{,}m_2\rangle \in \mathcal{DY}(M)}\ G_{\text{pair}}\,, \quad \frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{\{\!|m_1|\!\}_{m_2} \in \mathcal{DY}(M)}\ G_{\text{scrypt}}\,,$$

$$\frac{\langle m_1{,}m_2\rangle \in \mathcal{DY}(M)}{m_i \in \mathcal{DY}(M)}\ A_{\text{pair}_i}\,, \quad \frac{\{\!|m_1|\!\}_{m_2} \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{m_1 \in \mathcal{DY}(M)}\ A_{\text{scrypt}}\,.$$

The generation rules express that the intruder can compose messages from known messages using pairing and symmetric encryption, while the analysis rules describe how he can decompose messages.

The Dolev-Yao intruder leads to an enormous branching of the search tree when one naïvely enumerates all (meaningful) messages that the intruder can send. The lazy intruder technique significantly reduces the search tree without excluding any potential attacks, by exploiting the fact that the actual value of certain parts of a message is often irrelevant for the receiver. So, whenever the receiver will not further analyze the value of a particular message part, we can postpone during the search the decision about which value the intruder actually chooses for this part by replacing it with a variable and recording a *constraint* on which knowledge the intruder uses to generate the message. We express this information using constraints of the form $\mathit{from}(T, \mathit{IK})$, meaning that T is a set of terms generated by the intruder from the set of his known messages IK (for "intruder knowledge"). Appendix A contains a concrete example of the use of constraints in the analysis of the Yahalom protocol.

Definition 2. *A* from constraint *(or simply* constraint*) has the form $\mathit{from}(T, \mathit{IK})$, where T and IK are sets of terms. Its semantics is the set of satisfying ground substitutions σ for the variables in the constraint, i.e.*

$$[\![\mathit{from}(T, \mathit{IK})]\!] = \{\sigma \mid \mathrm{ground}(\sigma) \wedge \mathrm{ground}(T\sigma \cup \mathit{IK}\sigma) \wedge (T\sigma \subseteq \mathcal{DY}(\mathit{IK}\sigma))\}\,.$$

A constraint c is satisfiable *if $[\![c]\!] \neq \emptyset$. A (from)* constraint set *is a finite set of (from) constraints and its semantics is the intersection of the semantics of its elements modulo the different domains, i.e., overloading the $[\![\ldots]\!]$ notation, $[\![\{c_1, \ldots, c_n\}]\!] = \sqcap_{i=1}^{n} [\![c_i]\!]$. A constraint $\mathit{from}(T, \mathit{IK})$ is* simple*, written* $\mathrm{simple}(\mathit{from}(T, \mathit{IK}))$*, if $T \subseteq \mathcal{V}$. A constraint set is simple if all its constraints are simple. We overload* simple *and apply it to constraint sets as expected. A constraint set C is* well-formed *if one can index its constraints, $C = \{\mathit{from}(T_1, \mathit{IK}_1), \ldots, \mathit{from}(T_n, \mathit{IK}_n)\}$, so that the following conditions hold:* (1) *$\mathit{IK}_i \subseteq \mathit{IK}_j$ for $i \leq j$, and* (2) *$\mathrm{vars}(\mathit{IK}_i) \subseteq \cup_{j=1}^{i-1} \mathrm{vars}(T_j)$.*

Intuitively, (1) requires that the intruder knowledge increases monotonically and (2) requires that every variable that appears in intruder-known terms is part of a message that the intruder created earlier, i.e. variables only "originate" from the intruder. (As we explain in Appendix B, these properties are crucial for the completeness of the reduction functions *Red* and *D-Red* that we introduce below.) All constraint sets that we consider in the remainder of this paper are well-formed, unless stated otherwise.

The core of the lazy intruder technique is to reduce a given constraint set into an equivalent one that is either unsatisfiable or simple (and thus satisfiable, as it is straightforward to show that every simple constraint set is satisfiable since the intruder can always generate some message, e.g. simply by using his own name). In Figure 2 we give the generation and analysis rules of [5], which describe how constraint sets can be reduced (again, we consider only pairing and symmetric encryption and decryption for brevity). Note that here and elsewhere, we simplify notation for singletons, writing, e.g., $m_2 \cup \mathit{IK}$ for $\{m_2\} \cup \mathit{IK}$. A generation or analysis rule r has the form

$$\frac{C', \sigma'}{C, \sigma}\; r\,,$$

with C and C' constraint sets and σ and σ' substitutions, and it expresses that (C', σ') can be *derived* from (C, σ), which we denote by $(C, \sigma) \vdash_r (C', \sigma')$. That is, the constraint reduction rules are applied backwards. We extend the semantics function to such pairs, $[\![(C, \sigma)]\!] = \{\sigma\sigma' \mid \sigma' \in [\![C]\!]\}$, and we extend it pointwise to sets of such pairs.

$$\frac{\mathit{from}(m_1 \cup m_2 \cup M, IK) \cup C, \sigma}{\mathit{from}(\langle m_1,m_2\rangle \cup M, IK) \cup C, \sigma}\; G^L_{\text{pair}}\,, \quad \frac{\mathit{from}(m_1 \cup m_2 \cup M, IK) \cup C, \sigma}{\mathit{from}(\{\!|m_1|\!\}_{m_2} \cup M, IK) \cup C, \sigma}\; G^L_{\text{scrypt}}\,,$$

$$\frac{(\mathit{from}(M, m_2 \cup IK) \cup C)\tau, \sigma\tau}{\mathit{from}(m_1 \cup M, m_2 \cup IK) \cup C, \sigma}\; G^L_{\text{unif}}\; (\tau = \mathit{mgu}(m_1, m_2),\; m_1 \notin \mathcal{V})\,,$$

$$\frac{\mathit{from}(M, m_1 \cup m_2 \cup \langle m_1,m_2\rangle \cup IK) \cup C, \sigma}{\mathit{from}(M, \langle m_1,m_2\rangle \cup IK) \cup C, \sigma}\; A^L_{\text{pair}}\,(\{m_1, m_2\} \setminus IK \neq \emptyset),$$

$$\frac{\mathit{from}(m_2, IK) \cup \mathit{from}(M, m_1 \cup \{\!|m_1|\!\}_{m_2} \cup IK) \cup C, \sigma}{\mathit{from}(M, \{\!|m_1|\!\}_{m_2} \cup IK) \cup C, \sigma}\; A^L_{\text{scrypt}}\,(m_1 \notin IK).$$

Figure 2: Lazy intruder: constraint reduction rules

The generation rules G^L_{pair} and G^L_{scrypt} express that the constraint stating that the intruder can generate a message composed from submessages m_1 and m_2 using pairing or symmetric encryption can be replaced by the constraint stating that he can generate both m_1 and m_2. The rule G^L_{unif} expresses that the intruder can use a message m_2 from his knowledge if this message can be unified with the message m_1 he has to generate (note that both the terms to be generated and the terms in the intruder knowledge may contain variables). The reason that the intruder is "lazy" stems from the restriction that the G^L_{unif} rule cannot be applied when the term to be generated is a variable: the intruder's choice for this variable does not matter at this stage of the search and hence we postpone this choice.

The analysis of the intruder knowledge is complex as messages may contain variables and hence the possibility of the application of an analysis step may depend on the substitution for the variables. A straightforward way to express analysis in the lazy intruder setting is given by the rule A^L_{scrypt}: for a message $\{\!|m_1|\!\}_{m_2}$ that the intruder attempts to decrypt, we add the content m_1 to the intruder knowledge of the respective constraint (as if the check was already successful) and add a constraint expressing that the symmetric key m_2 needed for decryption must be generated from the same knowledge. This constraint renders the constraint set unsatisfiable when m_2 cannot be generated using the corresponding intruder knowledge.

Note that in the A^L_{scrypt} rule we also make the restriction that the message $\{\!|m_1|\!\}_{m_2}$ analyzed may not be used in the generation of the possibly non-atomic key m_2. This is in contrast to similar approaches that, like ours, can handle non-atomic symmetric keys, e.g. [23, 9]. The fact that our restriction does not exclude any solutions is shown as part of the proof of Theorem 1, the correctness and completeness of constraint reduction. Appendix B contains the completeness proof for the constraint reduction rules extended with the constraint differentiation technique given in §3, which includes as a special case the proof of Theorem 1 given in [5]. The proof in Appendix B also explains why removing the term $\{\!|m_1|\!\}_{m_2}$ from the knowledge to be used in the derivation of m_2 does not exclude any solutions.

Definition 3. *Let $\vdash$ denote the reflexive-transitive closure of the union of the derivation relations $\vdash_r$ for every rule r of Figure 2. The set of pairs of simple constraint sets and substitutions that can be derived from (C, id) is $\mathit{Red}(C) = \{(C', \sigma) \mid ((C, \mathrm{id}) \vdash (C', \sigma)) \wedge \mathrm{simple}(C')\}$. We call Red the* constraint reduction function*.*

Theorem 1. *Let C be a well-formed constraint set. $\mathit{Red}(C)$ is finite and $\vdash$ is well-founded. Moreover, $[\![C]\!] = [\![\mathit{Red}(C)]\!]$.*

By Theorem 1 we have that *Red* is correct, complete, and recursively computable.

2.2 Symbolic Transition Systems

The various implementations of the lazy intruder are based on different formalisms to represent the second layer of the search, namely the symbolic state space and its transition relation. For example, [5, 9] use multiset rewriting, [12, 23] use strand spaces, and [1, 7] use process calculi. Constraint differentiation is not specialized to any of these approaches, rather it only requires a state space where states are represented symbolically using terms with variables, together with a state transition function and a goal predicate. Therefore we define the concept of a symbolic transition system as an abstract interface to the particular symbolic formalism.

The idea behind symbolic approaches is to use symbolic states, i.e. terms with variables, to represent sets of ground states, i.e. sets of terms without variables. The permissible substitutions for variables are described by some kind of symbolic constraints, in our case *from* constraints. The semantics of a particular constraint set (i.e. the set of substitutions allowed by the constraints in the set) can then be extended to a symbolic state s (i.e. the set of ground terms represented by s). One can then define a state transition function as usual, but it must agree with the semantics of each symbolic state in the sense that equivalent symbolic states have equivalent successors and are equivalent with respect to a predicate, which in our case describes attacks. Formally:

Definition 4. *A* symbolic transition system *over a countable set Σ of* constant *and* function *symbols and a countable set $\mathcal{V}$ of variables is a 5-tuple $(\mathcal{G}, \mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{P})$, where $\mathcal{G} = T_\Sigma$ is the set of* ground states*; $\mathcal{S} = T_\Sigma(\mathcal{V}) \times \mathcal{CS}$ is the set of* symbolic states*, where $\mathcal{CS}$ denotes the set of all well-formed from constraint sets; $\mathcal{I} \in \mathcal{S}$ is the* initial symbolic state*; $\mathcal{T} : \mathcal{S} \to 2^{\mathcal{S}}$ is a* transition function *on symbolic states; and $\mathcal{P}$ is a predicate on symbolic states. We also refer to $\mathcal{G}$ as the* space of ground states*, and similarly for $\mathcal{S}$ and $\mathcal{CS}$, and call $\mathcal{P}$ the* attack predicate *as it will be used to specify symbolic states representing attacks. The semantics of a symbolic state $s = (t, C)$ is defined in terms of the semantics of the constraints:*

$$[\![(t, C)]\!] = \{t' \mid \exists \sigma.\ \sigma \in [\![C]\!] \wedge t' = t\sigma\}\,.$$

We straightforwardly extend $\mathcal{T}$, $\mathcal{P}$, and $[\![\cdot]\!]$ pointwise to sets of symbolic states. The symbolic transition system must agree with the semantics of the symbolic states in the following sense: for two sets of symbolic states S_1 and S_2 with $[\![S_1]\!] = [\![S_2]\!]$, we require that $[\![\mathcal{T}(S_1)]\!] = [\![\mathcal{T}(S_2)]\!]$ and $\mathcal{P}(S_1) \iff \mathcal{P}(S_2)$.

The set of reachable symbolic states *is the smallest set that contains the initial symbolic state and is closed under the transition function. A symbolic transition system is* secure *iff no reachable symbolic state satisfies the attack predicate.*

The constraint reduction is extended to symbolic states by

$$Red((t, C)) = \{(t', C') \mid \exists \sigma.\ (C', \sigma) \in Red(C)\ \wedge t' = t\sigma\}\,,$$

so that we straightforwardly have that $[\![Red(s)]\!] = [\![s]\!]$ for every symbolic state s.

The other lazy intruder approaches can be easily recast as symbolic transition systems. For example, in the concrete case of the model underlying the tool OFMC [5], the ground states in $\mathcal{G}$ are multisets of facts which express the local state of agents, the intruder knowledge, and the messages sent on the network but not yet received. The symbolic states in $\mathcal{S}$ are also multisets of facts, but message terms may contain variables; hence, a symbolic state represents the set of ground states that are obtained by a ground substitution for the variables. The transition function on symbolic states $\mathcal{T}$ is induced by multiset rewrite rules that describe the behavior of the honest agents, and add constraints to the constraint set when the intruder has to generate a new message (note that by the construction of this transition function it is ensured that in all reachable symbolic states the constraint set is well-formed). Finally, the attack predicate $\mathcal{P}$ is used to express state-based properties on symbolic states. In OFMC we use $\mathcal{P}$ to formulate standard authentication and secrecy goals, but in the abstract symbolic transition system above we commit neither to particular kinds of attacks nor to a particular formalism to specify attacks (any of the other formalisms previously listed, e.g. strands, could be used), as long as they are formalized as reachability problems.

A symbolic transition system gives rise to a search tree where the root node is the initial state and the children of a node are all states that can be reached with one transition. Applying the function *Red* to every symbolic state yields a set of equivalent symbolic states with simple constraint sets. This can be exploited for a reduction since if the constraint set C of a symbolic state is unsatisfiable, then $Red(C) = \emptyset$ by Theorem 1. In this case, we can safely prune the subtree of the search tree node containing the unsatisfiable constraint. In the next section, we will see that the integration of constraint differentiation into the symbolic transition system is based on a similar form of pruning of the search tree.

Before considering this integration in detail, let us first observe that the lazy intruder can be straightforwardly extended with a technique that we call *step-compression*, which leads to a significant reduction of the search tree without excluding any attacks.[2] Step-compression is based on the idea that since the intruder completely controls the communication network, we can safely assume that every message from an honest agent is automatically intercepted by the intruder (who can always play it back into the network) and that every message that an honest agent receives comes from the intruder. This allows us to restrict the search to transitions where two steps are merged (or "compressed") into one: first, the intruder sends a message to an honest agent and second, the intruder intercepts the agent's reply.

When step-compression is used, the symbolic transition system has the following property: for every transition from a symbolic state $s_1 = (t_1, C_1)$ to a symbolic state $s_2 = (t_2, C_2)$, the constraint sets will have the form $C_2 = C_1 \cup from(m_1, IK)$ for some message m_1, representing the message the intruder sends to an honest agent, and a set of messages IK, representing the knowledge the intruder can use to generate m_1. Also, the intruder knowledge in s_2 is augmented by the agent's reply. We will make use of this property in the constraint differentiation technique.

3 Constraint Differentiation

The lazy intruder technique allows us to significantly reduce the search tree generated by the prolific Dolev-Yao intruder without excluding any attacks (cf. [5]). In particular, our experiments have shown that the search tree induced by the lazy intruder often has roughly the same size as the tree that would be searched when considering a *passive* intruder, i.e. one that listens to the communication on the network but does not manipulate or generate any messages. This is the maximal reduction possible since the search tree of reachable symbolic states must cover all "legal" executions of the protocol between honest agents.

However, even when considering only a small number of sessions that can be executed in parallel, searching the tree that contains all the interleavings of these sessions can be infeasible; we are faced with the standard state-explosion problem of model-checking. In model-checking approaches for concurrent systems, this problem is often handled using *partial-order reduction* (POR), a technique that reduces the number of interleavings that need to be considered by exploiting independencies between the possible transitions [26].

[2] Note that step-compression is applied not only in all symbolic approaches we know of [1, 5, 6, 7, 8, 12, 15, 17, 23], but also in some non-symbolic approaches, e.g. [2].

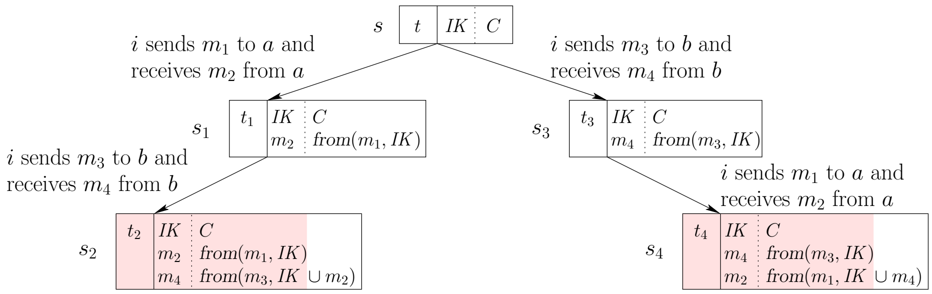

Figure 3: An illustration of constraint differentiation for $t_2 = t_4$ (for each symbolic state s we display t and C as well as the corresponding intruder knowledge IK).

One might expect that the direct combination of the lazy intruder with partial-order reduction would allow us to simultaneously "solve" both state-explosion problems, that of the prolific Dolev-Yao intruder and that of the large number of possible interleavings. However, their direct combination is not effective: the different transitions of the lazy intruder rarely lead to the same successor state and therefore there is practically no independence of transitions that can be exploited by POR. We present a way around this problem by directly using independence information in the constraint reduction; the result is a POR-inspired reduction technique that we call *constraint differentiation*.

To see why the direct combination of partial-order reduction with the lazy intruder is not effective, consider the successor function of the search tree that results from the symbolic transition system (with step-compression) given in the previous section. A direct application of POR would require identifying situations of the form depicted in Figure 3. There are two sequences of transitions. In the left one, the intruder i first sends a message m_1 to an agent a, receiving the answer m_2, and afterwards he sends a message m_3 to an agent b, receiving the answer m_4. In the right sequence, the intruder first talks to b and then to a. The transitions result in states $s_2 = (t_2, C_2)$ and $s_4 = (t_4, C_4)$, for terms (with variables) t_2 and t_4 and constraint sets C_2 and C_4. We consider the case $t_2 = t_4$, which holds when the transitions are independent in the sense that on ground states the respective order of operations would lead to the same successor states. In this case, for every substitution σ, the represented ground states $t_2\sigma$ and $t_4\sigma$ are the same, while the constraints, determining the set of permissible substitutions, are different due to the fact that the intruder generated the respective messages at different states of his knowledge. Hence, the direct combination of partial-order reduction with the lazy intruder is not effective.[3]

The observation that leads to a reduction is that in this situation there is an overlapping of the set of ground states, which is represented by the two symbolic states s_2 and s_4, as shown by the shaded part in Figure 3: all those ground states in the semantics of the symbolic states that do not exploit the new knowledge m_2 or m_4, respectively, are covered by the other symbolic state. The idea is that we can use independence of transitions by exploiting precisely this overlap. If, for example, we "prefer" the left sequence, then for the state s_4 reached by the other sequence we will only be interested in solutions that are not subsumed by s_2 already, i.e. those where the intruder actually uses the message m_4 that he learned in the first transition to generate the message m_1 in the second transition.

[3]One may wonder whether without step-compression, i.e. without this property of the constraints, POR could be more effective. One can show that directly applying POR to a symbolic transition system without step-compression can only result in reductions that are also achieved by step-compression. This means that it is without loss of generality to integrate step-compression in the symbolic transition system.

In this way, we can propagate information about independencies obtained on the second search layer, the symbolic transition system, to the first layer, the constraint reduction. The information we exploit is the fact that we only need to consider solutions for a given constraint set that are obtained by using new knowledge. In the example, we could express the fact that the message m_4 needs to be used when creating m_1 by using constraints of the form $\mathit{D\text{-}from}(m_1, \mathit{IK}, m_4)$, which intuitively has the same meaning as the constraint $\mathit{from}(m_1, \mathit{IK} \cup m_4)$, except that we exclude all solutions of $\mathit{from}(m_1, \mathit{IK})$.

Mirroring the development of §2, we proceed as follows: we introduce constraint differentiation by defining this new kind of constraint (D-from constraints), the associated reduction rules (D-from rules), and the constraint reduction function (*D-Red*) that describes the constraints that can be derived with these rules. We then show that the *D-Red* function has properties analogous to those of the *Red* function, namely it is correct, complete, and recursively computable. We conclude the section by integrating the new constraint reduction into the second layer of the search as a transformation of the search tree induced by the symbolic transition system.

3.1 Constraint Reduction with Constraint Differentiation

We now extend Definition 4 of a symbolic transition system with the following form of constraints.

Definition 5. *A* D-from constraint *c has the form $\mathit{D\text{-}from}(T, \mathit{IK}, \mathit{NIK})$, where T, IK, and NIK are sets of messages. Its semantics is*

$$[\![c]\!] = [\![\lceil c \rceil]\!] \setminus [\![\lfloor c \rfloor]\!] \,,$$

where $\lceil \mathit{D\text{-}from}(T, \mathit{IK}, \mathit{NIK}) \rceil = \mathit{from}(T, \mathit{IK} \cup \mathit{NIK})$ and $\lfloor \mathit{D\text{-}from}(T, \mathit{IK}, \mathit{NIK}) \rfloor = \mathit{from}(T, \mathit{IK})$ are functions mapping D-from constraints to from constraints. We say that $\mathit{D\text{-}from}(T, \mathit{IK}, \mathit{NIK})$ is simple *if $T \subseteq \mathcal{V}$ and $T \neq \emptyset$.*

Extending Definition 2, we now write simply constraint *to refer to a from constraint or a D-from constraint. The semantics of a constraint set $C = \{c_1, \ldots, c_n\}$ is $[\![\{c_1, \ldots, c_n\}]\!] = \sqcap_{i=0}^{n} [\![c_i]\!]$. The definitions of $\lceil \cdot \rceil$, $\lfloor \cdot \rfloor$, and* simple *are extended straightforwardly to constraint sets, and a constraint set C is* well-formed *iff both $\lceil C \rceil$ and $\lfloor C \rfloor$ are well-formed.*

Intuitively, NIK represents new messages that are not in IK (it is an acronym for *new intruder knowledge*). As we will use them, NIK and IK will be disjoint, so that the constraint $\mathit{D\text{-}from}(T, \mathit{IK}, \mathit{NIK})$ states that the set of terms T must be generated by the intruder using the knowledge in the set $\mathit{IK} \cup \mathit{NIK}$, but we are only interested in solutions that employ new information in NIK; hence, we exclude all solutions of $\mathit{from}(T, \mathit{IK})$. Note that *from* constraints are a special case of *D-from* constraints since $\mathit{from}(T, \mathit{IK}) = \mathit{D\text{-}from}(T, \emptyset, \mathit{IK})$. The function $\lceil \cdot \rceil$ yields a from constraint set by removing the requirement on new knowledge; therefore $[\![C]\!] \subseteq [\![\lceil C \rceil]\!]$. Similarly, $[\![\lfloor C \rfloor]\!] \cup [\![C]\!] = [\![\lceil C \rceil]\!]$, as $\lfloor \cdot \rfloor$ returns the solutions removed from $\lceil C \rceil$. Note that for a simple constraint set C both $\lceil C \rceil$ and $\lfloor C \rfloor$ are simple (and hence satisfiable). Note that if a constraint set C is simple then $\lceil C \rceil$ is satisfiable.[4]

Let us extend the set of lazy intruder reduction rules of Figure 2 with the reduction rules for the new D-from constraints, for short *D-from rules*, which we display in Figure 4. (Appendix A contains an example of the use of D-from constraints and their reduction.) As representative cases, we only explain the three most interesting D-from rules, i.e. G^{LD}_{unif2}, $A^{LD}_{\mathrm{scrypt2}}$, and $A^{LD}_{\mathrm{scrypt3}}$. The rules

[4]Note also that, unlike for from constraints, it does not always hold that every simple D-from constraint set C is satisfiable (although this is usually the case). However, if C is simple then $\lceil C \rceil$ is satisfiable. Hence, not excluding C from the search is not a problem of correctness but only of efficiency.

$$\frac{\textit{D-from}(m_1 \cup m_2 \cup M, IK, NIK) \cup C, \sigma}{\textit{D-from}(\langle m_1,m_2\rangle \cup M, IK, NIK) \cup C, \sigma}\; G^{LD}_{\text{pair}}\,, \quad \frac{\textit{D-from}(m_1 \cup m_2 \cup M, IK, NIK) \cup C, \sigma}{\textit{D-from}(\{\!|m_1|\!\}_{m_2} \cup M, IK, NIK) \cup C, \sigma}\; G^{LD}_{\text{scrypt}}\,,$$

$$\frac{(\textit{D-from}(M, m_2 \cup IK, NIK) \cup C)\tau, \sigma\tau}{\textit{D-from}(m_1 \cup M, m_2 \cup IK, NIK) \cup C, \sigma}\; G^{LD}_{\text{unif1}}\ (\tau = mgu(m_1, m_2), m_1 \notin \mathcal{V})\,,$$

$$\frac{(\textit{from}(M, m_2 \cup IK \cup NIK) \cup C)\tau, \sigma\tau}{\textit{D-from}(m_1 \cup M, IK, m_2 \cup NIK) \cup C, \sigma}\; G^{LD}_{\text{unif2}}\ (\tau = mgu(m_1, m_2), m_1 \notin \mathcal{V})\,,$$

$$\frac{\textit{D-from}(M, m_1 \cup m_2 \cup \langle m_1,m_2\rangle \cup IK, NIK) \cup C, \sigma}{\textit{D-from}(M, \langle m_1,m_2\rangle \cup IK, NIK) \cup C, \sigma}\; A^{LD}_{\text{pair1}}\,(\{m_1, m_2\} \setminus IK \neq \emptyset),$$

$$\frac{\textit{D-from}(M, IK, \langle m_1,m_2\rangle \cup M' \cup NIK) \cup C, \sigma}{\textit{D-from}(M, IK, \langle m_1,m_2\rangle \cup NIK) \cup C, \sigma}\; A^{LD}_{\text{pair2}}\ (M' = \{m_1, m_2\} \setminus IK, \{m_1, m_2\} \setminus (IK \cup NIK) \neq \emptyset)\,,$$

$$\frac{\textit{D-from}(M, m_1 \cup \{\!|m_1|\!\}_{m_2} \cup IK, NIK) \cup \textit{from}(m_2, IK) \cup C, \sigma}{\textit{D-from}(M, \{\!|m_1|\!\}_{m_2} \cup IK, NIK) \cup C, \sigma}\; A^{LD}_{\text{scrypt1}}\,(m_1 \notin IK),$$

$$\frac{\textit{D-from}(M, \{\!|m_1|\!\}_{m_2} \cup IK, m_1 \cup NIK) \cup \textit{D-from}(m_2, IK, NIK) \cup C, \sigma}{\textit{D-from}(M, \{\!|m_1|\!\}_{m_2} \cup IK, NIK) \cup C, \sigma}\; A^{LD}_{\text{scrypt2}}\ (m_1 \notin IK)\,,$$

$$\frac{\textit{D-from}(M, IK, m_1 \cup \{\!|m_1|\!\}_{m_2} \cup NIK) \cup \textit{from}(m_2, IK \cup NIK) \cup C, \sigma}{\textit{D-from}(M, IK, \{\!|m_1|\!\}_{m_2} \cup NIK) \cup C, \sigma}\; A^{LD}_{\text{scrypt3}}\ (m_1 \notin (IK \cup NIK))\,.$$

Figure 4: The D-from rules for generation and analysis

for the analysis of pairs are similar to those for the analysis of encryption and the other rules are simply 'liftings' of the original rules to D-from constraints.

As shorthand, let us use the terms *T-part*, *IK-part*, and *NIK-part* to refer to the respective parts of a D-from constraint $\textit{D-from}(T, IK, NIK)$. The G^{LD}_{unif2} rule expresses the case where the term m_1, that should be generated, can be unified with a term m_2 in the NIK-part. Unless m_1 can be derived from the IK-part, the condition that some message from the NIK-part must be used is satisfied, and thus we are not obliged to use messages in NIK for generating further terms in the T-part. However, if m_1 can be derived from IK, then the application of the rule is inefficient (as it removes a possible restriction), but still correct in the sense that the resulting solution is valid.

The A^{LD}_{scrypt2} rule describes the analysis of an encrypted term $\{\!|m_1|\!\}_{m_2}$ of the IK-part, where m_1 is not yet in the IK-part of the D-from constraint. Suppose the key m_2 can be derived using the intruder knowledge $IK \cup NIK$ but not from IK alone, then m_1 can only be derived when using knowledge in NIK. Therefore using m_1 itself means that knowledge from NIK is (indirectly) used (unless there is another analyzable term from which m_1 can be derived, but this is, as in the previous case, only a matter of efficiency, not of correctness). Therefore we have the newly learned term m_1 in the NIK-part and the constraint that m_2 can be derived only when using NIK.

The A^{LD}_{scrypt3} rule describes, similarly, the case that the term to analyze is in the NIK-part. In this case, the result of the decryption is also added to the NIK-part, as the analysis is based on terms in NIK and could not be performed without NIK. However, to analyze the key m_2 we need not use information in NIK, so the generation of m_2 is expressed with a from constraint.

Definition 6. *Let $\vdash^D$ be the extension of the derivation relation $\vdash$ (cf. Definition 3) with the rules in Figure 4. The set of pairs of simple D-from constraint sets and substitutions that can be derived from (C, id) is $\textit{D-Red}(C) = \{(C', \sigma) \mid ((C, \mathrm{id}) \vdash^D (C', \sigma)) \wedge \mathrm{simple}(C')\}$.*

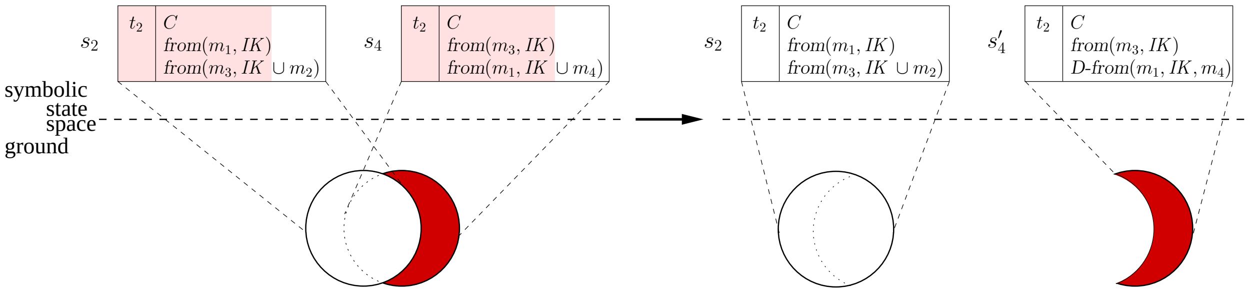

Figure 5: Constraint differentiation at work.

3.2 Properties of *D-Red*

In this section, we show that the *D-Red* function has analogous properties to the *Red* function, namely correctness, completeness, and termination. Analogously to Theorem 1, we would like to have the property $[\![C]\!] = [\![D\text{-}Red(C)]\!]$. However, this does not hold: there may be solutions of $(C', \sigma) \in D\text{-}Red(C)$ that are not subsumed by C (but by $\lceil C \rceil$). An example is the D-from constraint $C = D\text{-}from(m, k \cup \{\!|m|\!\}_k, m)$. Obviously, the intruder can derive the term m without the new knowledge, hence C is unsatisfiable. However, the rule G^{LD}_{unif2} is applicable, leading to a from constraint with an empty T-part: $C' = from(\emptyset, k \cup \{\!|m|\!\}_k \cup m)$. As C' is simple, $[\![C']\!] = \{\text{id}\}$. Note that although this solution is not contained in $[\![C]\!]$, it is contained in $[\![\lceil C \rceil]\!]$, meaning that *D-Red* returns more solutions than desired, but still correct solutions — it has missed a possible reduction. This is a problem of efficiency, not of correctness and completeness. Hence we relax the above statement and show correctness only with respect to the original approach (the proofs can be found in Appendix B):

Theorem 2. *Let C be a well-formed constraint set. D-Red(C) is finite and $\vdash^D$ is well-founded. Moreover, $[\![D\text{-}Red(C)]\!] \subseteq [\![\lceil C \rceil]\!]$.*

For the other direction, i.e. completeness, we have that all solutions of C are contained in *D-Red*:

Theorem 3. *$[\![C]\!] \subseteq [\![D\text{-}Red(C)]\!]$ for a well-formed constraint set C.*

These theorems tell us that *D-Red* is correct, complete, and recursively computable. This implies that we can apply *D-Red* to reduce the search space and thereby neither exclude attacks nor introduce new ones. To formalize this reduction of the search space, we integrate constraint differentiation into the search tree induced by the symbolic transition system.

3.3 Integration of Constraint Differentiation into Symbolic Transition Systems

Consider again the tree of Figure 3. This situation (which occurs whenever two independent actions are performed in two different orders, as explained above) characterizes when we apply constraint differentiation: we exploit the fact that the two symbolic states s_2 and s_4 of Figure 3 represent overlapping sets of ground states as shown by the shaded parts in s_2 and s_4.

Figure 5 merges parts of Figures 1 and 3 to illustrate how constraint differentiation works: we pick one, say s_4, of the overlapping states s_2 and s_4 in Figure 3 (where $t_2 = t_4$) and replace the *from* constraint that does not appear in the other constraint set with a *D-from* constraint; this yields the transformed state s'_4. That is, we use constraint *differentiation* to restrict the extension of one of the two symbolic states to those ground states that are not covered by the other (as illustrated by the shaded part in the set of ground states). The following theorem shows that the two states s_2 and s_4' represent the same ground states as s_2 and s_4.

Theorem 4. *Let $s_2 = (t_2, C_2)$ and $s_4 = (t_2, C_4)$ be symbolic states with constraint sets of the form $C_2 = C \cup \mathit{from}(m_1, \mathit{IK}) \cup \mathit{from}(m_3, \mathit{IK} \cup m_2)$ and $C_4 = C \cup \mathit{from}(m_3, \mathit{IK}) \cup \mathit{from}(m_1, \mathit{IK} \cup m_4)$ for some messages m_1, m_2, m_3, m_4 and some constraint set C. Then*

$$[\![s_2]\!] \cup [\![s_4]\!] = [\![s_2]\!] \cup [\![s_4']\!]$$

for $C_4' = C \cup \mathit{from}(m_3, \mathit{IK}) \cup D\text{-}\mathit{from}(m_1, \mathit{IK}, m_4)$ and $s_4' = (t_2, C_4')$.

Proof. It is sufficient to show that $[\![C_2]\!] \cup [\![C_4]\!] = [\![C_2]\!] \cup [\![C_4']\!]$. Since $\lceil C_4' \rceil = C_4$ and $[\![\lceil C_4' \rceil]\!] \supseteq [\![C_4']\!]$, the direction $\supseteq$ is trivial. To show $[\![C_2]\!] \cup [\![C_4]\!] \subseteq [\![C_2]\!] \cup [\![C_4']\!]$, it is sufficient to show that for every solution $\sigma \in [\![C_4]\!]$ with $\sigma \notin [\![C_2]\!]$ it holds that $\sigma \in [\![C_4']\!]$. Suppose $\sigma \notin [\![C_4']\!]$. Then, since $\sigma \in [\![C_4]\!]$, it must hold that $\sigma \in [\![C_4]\!] \setminus [\![C_4']\!]$, i.e. $\sigma \in [\![\lfloor C_4 \rfloor]\!]$. Hence, $\sigma \in [\![\mathit{from}(m_1, \mathit{IK})]\!]$ and, since $\sigma \in [\![\mathit{from}(m_3, \mathit{IK})]\!]$, we have $\sigma \in [\![C_2]\!]$, which contradicts the assumption. □

This theorem allows us to perform a transformation on the search tree that replaces from constraints with more restrictive D-from constraints without changing the set of represented ground states. If under the more restrictive constraint C_4' the intruder could not use any new message from his knowledge, then even if C_4 is satisfiable, C_4' is unsatisfiable (so that the shaded part of the set of ground states in Figure 5 is also empty), which we can check using *D-Red*. This is the maximal reduction that can be achieved by constraint differentiation: the node of the state s_4 and its subtree can be completely removed from the search tree as the intruder could not generate anything "interesting", i.e. nothing that he could not have generated before. Note that, by symmetry, we can always consider the inverse situation, i.e. if performing the restriction on s_2 rather than s_4 leads to an unsatisfiable constraint set, then we can remove the respective subtree.

When we apply *D-Red* to a state that results by replacing from constraints with D-from constraints, in the best case the constraint set turns out to be unsatisfiable, so the state (and the respective subtree) can be removed. However, it is also possible that after applying *D-Red* there still remain simple D-from constraints (i.e. with variables in the T-part). This means that it is not yet determined what the intruder will use here, so it is possible that it is some message from NIK. Such a D-from constraint is nonetheless useful for the reduction, as it constrains the child nodes by storing that certain solutions can be excluded: the D-from constraint prevents all later instantiations of the variable in the T-part if these instantiations do not use some message of the NIK-part.

4 Experimental Results

To test the practical relevance of constraint differentiation, we have implemented it into the on-the-fly protocol model-checker OFMC, which is based on the lazy infinite-state approach to protocol analysis presented in [4]. In this approach, the search space can be represented as a potentially infinite tree that is generated on demand.[5] We have straightforwardly implemented constraint differentiation as a filter that is applied to, and prunes, the search tree. We now describe experiments which show that constraint differentiation significantly improves both the performance and the scope of OFMC.

[5] There is no relation between the lazy intruder and the lazy protocol analysis of [4], except that both are based on demand-driven evaluation.

Even without constraint differentiation, OFMC is a state-of-the-art tool for finding protocol flaws, as shown in [5]: on a 2,4 GHz Pentium-4 PC (with 512 MB RAM, but OFMC is not memory intensive), OFMC takes 5.29 seconds of cumulative CPU time to detect flaws in all of the 35 protocols of the Clark-Jacob-library [10] that are known to be flawed [14]. OFMC also discovered a previously unknown flaw in the Yahalom protocol (see also Appendix A). Moreover, we have applied OFMC to large-scale protocols including IKE, SET, CHAP, and various other industrial protocols currently being standardized by the Internet Engineering Task Force [18].[6] In particular, in [5] we describe our analysis of the H.530 protocol [19], a protocol developed by Siemens and proposed as an Internet standard for multimedia communications. The developers of the protocol performed an analysis using the finite-state model-checker Casper/FDR [14, 21], which ran out of memory due to the complexity of the protocol. We have modeled H.530 in its full complexity and have detected a simple replay attack in 1.62 seconds. The weakness is serious enough that Siemens has changed the protocol.

The integration of constraint differentiation has allowed us to improve the performance and extend the scope of OFMC so that it not only scales well to the falsification of industrial-strength protocols, but it can also be applied as an effective verification tool, exhaustively searching the large space associated with bounded numbers of protocol sessions. We have carried out a large number of experiments to validate our approach.

As a first example, using constraint differentiation the total time for detecting the flaws in the Clark-Jacob library drops from $5.29s$ to $3.36s$, and the detection of the H.530 flaw drops from $1.62s$ to $0.80s$. The improvements achieved by constraint differentiation are more dramatic the more complex the analyzed protocols and their flaws are, and when performing verification (for a bounded number of sessions). Verifying correct protocols is usually substantially more complex than falsifying flawed protocols, since flaw detection terminates as soon as an error is detected, while for verification the entire search space must be examined. Constraint differentiation has enabled us to perform verification for considerably larger numbers of parallel sessions than was possible with the previous version of the tool.

As a relevant example of an industrial-strength protocol we discuss here our analysis of the protocol-suite IKE [16]. We have used OFMC to analyze the full specification of each of the individual subprotocols of IKE and several combinations of them. By "full" we mean that we did not simplify the structure of the messages, which contain highly complex key-terms. OFMC detected a number of minor weaknesses of IKE, which have also been reported in [22, 30]. To discuss concrete performance results, we consider two subprotocols of IKE, the *Main Mode* and the *Aggressive Mode* of Phase 1 in the pre-shared key variants; the running times of the other subprotocols are similar.

Figure 6 compares the size of plies of the search tree for Aggressive Mode without and with constraint differentiation. We consider six scenarios varying in the number of parallel and consecutive sessions. We consider two ($[a, b], [a, i]$), three (also $[i, a]$) or four (also $[b, i]$) sessions, where $[a, b]$ means that agent a plays the protocol initiator role and agent b the responder role, and i is the intruder.[7] Moreover, sn denotes n consecutive sessions (where s1, i.e. only one session, is the standard case), meaning that an honest agent who has finished his part of the protocol session (for the jth time, respectively) is prepared to engage in $n-1$ ($n-j$, respectively) additional consecutive

[6] Several of these protocols employ the Diffie-Hellman key-exchange, which requires relaxing the free algebra assumption. Discussing how we realized this is beyond the scope of this paper.

[7] Such an explicit declaration of the intruder is only necessary when he shall participate under his real name, modeling a compromised or dishonest agent, while he can always, without explicit declaration, send messages under an arbitrary identity.

IKE Aggressive Mode Pre-Shared Key												
Mode:	without CD						with CD					
Scenario:	$[a,b],[a,i]$		$[a,b],[a,i],[i,a]$		$[a,b],[a,i],[i,a],[b,i]$		$[a,b],[a,i]$		$[a,b],[a,i],[i,a]$		$[a,b],[a,i],[i,a],[b,i]$	
Ply	s1	s2	s1	s2	s1	s2	s1	s2	s1	s2	s1	s2
1	3	3	4	4	5	5	3	3	4	4	5	5
2	7	7	14	14	23	23	5	5	10	10	16	16
3	13	14	43	45	97	100	7	8	19	21	40	43
4	17	27	112	139	368	420	6	12	30	44	86	111
5	15	53	238	422	1228	1727	5	17	35	81	150	261
6	15	101	393	1262	3501	6989	3	18	31	139	218	578
7		191	483	3699	8232	27835		20	22	215	241	1174
8		410	420	10637	15288	108927		23	8	319	203	2290
9		720		29783	21168	417862		22		436	136	4112
10		960		79939	18900	1565354		12		527	48	7025
11		990		201861		5695140		9		602		11062
12		990		467533		TO		5		576		16390
13				929500		TO				428		22544
14				1583582		TO				233		27443
15				2132130		TO				177		31024
16				1801800		TO				53		29595
17						TO						10531
18						TO						10531
19						TO						7857
20						TO						2371
Nodes	71	4467	1708	7242353	68811	TO	30	155	160	3866	1144	197426
Time	0.16s	13.66s	4.64s	40655.50s	3m41s	TO	0.08s	0.49s	0.49s	21.60s	4.17s	26m30s

Figure 6: Comparison of OFMC without and with constraint differentiation (CD) for IKE Aggressive Mode Pre-Shared Key: the nodes for each ply of the search tree and search time.

sessions with the same partners. Consecutive sessions are valuable for detecting replay attacks as they create a smaller search space than the same sessions in parallel. We display the number of nodes on each ply of the search tree; note that for the “smaller” scenarios (i.e. fewer parallel and consecutive sessions) the depth of the search tree is smaller, hence the empty cells. We also display the total number of nodes of the tree and the CPU time for searching the entire tree (TO denotes *time out* after one day, i.e. 1440 minutes of CPU time).

Figure 6 shows that constraint differentiation is most effective, as measured by the number of nodes that must be searched, when the original search space contains many interleavings of parallel sessions. The savings are most dramatic on the deeper plies of the search tree as the number of interleavings grows exponentially in the original model; since many interleavings are redundant and constraint differentiation can exploit this redundancy, the number of nodes does not necessarily grow exponentially with the depth of the tree. The difference between an exponential growth without constraint differentiation and an often sub-exponential growth with constraint differentiation leads to more dramatic savings the deeper the tree is searched. Note also that without constraint differentiation the number of nodes on each ply typically grows monotonically with the depth of the ply; with constraint differentiation the number grows on the first few plies and then starts to shrink again. This phenomenon is explained by the fact that with constraint differentiation the deeper we are in the tree, the more successor nodes are completely excluded. (The intuition behind this is that many transitions possible in the original model do not permit the use of newly learned messages and are thus pruned from the tree by constraint differentiation.)

In summary, by employing constraint differentiation, OFMC scales significantly better with the size of the considered scenario. This both extends the scope of OFMC and allows its use in verification for bounded scenarios. Figure 7 shows the broader picture, assessing the savings due to constraint differentiation for both main and aggressive mode for different parallel and consecutive sessions. Constraint differentiation reduces the search space significantly in all cases, and in some cases it even enables the analysis of problems that were out of the scope of OFMC and other tools.

Scenario	sn	IKE Main Mode Pre-Shared Key					IKE Aggressive Mode Pre-Shared Key				
		Ply	without CD		with CD		Ply	without CD		with CD	
			Time	Nodes	Time	Nodes		Time	Nodes	Time	Nodes
$[a, b]$	1	7	0.11	36	0.08	24	4	0.02	7	0.01	6
	2	14	0.30	91	0.20	61	8	0.04	18	0.02	12
$[a, b], [a, i]$	1	11	45.41	8969	9.18	1902	6	0.15	71	0.08	30
	2	22	TO	TO	11m25	98271	12	13.82	4467	0.48	155
$[a, b], [i, b]$	1	10	10.99	2034	3.93	704	6	0.18	75	0.08	28
	2	20	185m52	1014127	3m45	21155	12	18.51	4897	0.57	159
$[a, b], [a, b]$	1	14	22m43	169531	2m38	19642	8	1.81	689	0.35	123
	2	28	TO	TO	429m09	2020491	16	68m56	906789	6.65	1508

Figure 7: Comparison of search time and tree size without and with constraint differentiation (CD) for IKE Main Mode and Aggressive Mode Pre-Shared Key.

5 Conclusions and Future Work

Constraint differentiation effectively integrates the lazy intruder with ideas from partial-order reduction. We have proven that this integration does not change the set of represented ground states and hence is correct and complete. We have shown too, empirically, that it leads to dramatic reductions in the size of the state space searched, considerably improving the performance of the model-checker OFMC and making possible new kinds of analysis, i.e. both falsification of industrial-strength protocols and verification of bounded numbers of sessions of such protocols.

We see room for further improvements of the constraint differentiation technique. One promising idea is the use of the *D-from* constraints for heuristic search by focusing on "interesting" protocol interleavings, i.e. ones where at every step the intruder uses knowledge obtained during the previous step. We would also like to exploit results concerning bounds on the number of agents and parallel sessions that need to be considered for protocol verification, e.g. [29, 11].

References

1. R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. INRIA Research Report 3915, 2000.

2. A. Armando and L. Compagna. Automatic SAT-Compilation of Protocol Insecurity Problems via Reduction to Planning. In *Proceedings of FORTE 2002*, LNCS 2529, pages 210–225. Springer-Verlag, 2002.

3. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

4. D. Basin. Lazy infinite-state analysis of security protocols. In *Proceedings of CQRE'99*, LNCS 1740, pages 30–42. Springer-Verlag, 1999.

5. D. Basin, S. Mödersheim, and L. Viganò. An On-The-Fly Model-Checker for Security Protocol Analysis. Technical Report 404, Dep. of Computer Science, ETH Zurich, 2003. Available at `www.inf.ethz.ch/research/publications/`.

6. M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proceedings of ICALP'01*, LNCS 2076, pages 667–681. Springer-Verlag, 2001.

7. M. Boreale and M. G. Buscemi. A framework for the analysis of security protocols. In *Proceedings of CONCUR 2002*, LNCS 2421, pages 483–498. Springer-Verlag, 2002.

8. Y. Chevalier and L. Vigneron. A Tool for Lazy Verification of Security Protocols. In *Proceedings of ASE'01*. IEEE Computer Society Press, 2001.

9. Y. Chevalier and L. Vigneron. Automated Unbounded Verification of Security Protocols. In E. Brinksma and K. G. Larsen, editors, *Proceedings of CAV'02*, LNCS 2404, pages 324–337. Springer-Verlag, 2002.

10. J. Clark and J. Jacob. A Survey of Authentication Protocol Literature: Version 1.0, 17. Nov. 1997. URL: `www.cs.york.ac.uk/~jac/papers/drareview.ps.gz`.

11. H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. In *Proc. 12th European Symposium on Programming (ESOP'2003)*, volume 2618 of *LNCS*, pages 99–113, 2003.

12. R. Corin and S. Etalle. An Improved Constraint-Based System for the Verification of Security Protocols. In *Proceedings of SAS 2002*, LNCS 2477, pages 326–341. Springer-Verlag, 2002.

13. D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.

14. B. Donovan, P. Norris, and G. Lowe. Analyzing a Library of Security Protocols using Casper and FDR. In *Proceedings of the Workshop on Formal Methods and Security Protocols*, 1999.

15. M. Fiore and M. Abadi. Computing Symbolic Models for Verifying Cryptographic Protocols. In *Proceedings of CSFW'01*. IEEE Computer Society Press, 2001.

16. D. Harkins and D. Carrel. RFC 2409: The Internet Key Exchange (IKE). 1998.

17. A. Huima. Efficient infinite-state analysis of security protocols. In *Proceedings of the FLOC'99 Workshop on Formal Methods and Security Protocols (FMSP'99)*, 1999.

18. IETF: The Internet Engineering Task Force.

19. ITU-T Recommendation H.530: Symmetric Security Procedures for H.510 (Mobility for H.323 Multimedia Systems and Services). 2002.

20. G. Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop: CSFW'97*, pages 31–43. IEEE Computer Society Press, 1997.

21. G. Lowe. Casper: a Compiler for the Analysis of Security Protocols. *Journal of Computer Security*, 6(1):53–84, 1998. See `http://web.comlab.ox.ac.uk/oucl/work/gavin.lowe/Security/Casper/`.

22. C. Meadows. Analysis of the Internet Key Exchange Protocol Using the NRL Protocol Analyzer. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1999.

23. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 166–175, 2001.

24. J. C. Mitchell, M. Mitchell, and U. Stern. Automated Analysis of Cryptographic Protocols Using Murphi. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 141–153, 1997.

25. L. C. Paulson. Relations between secrets: The Yahalom protocol. In *Proceedings of the 7th Cambridge International Workshop on Security Protocols*, LNCS 1796, pages 73–77. Springer-Verlag, 1999.

26. D. Peled. Ten Years of Partial Order Reduction. In *Proceedings of CAV 1998*, LNCS 1427, pages 17–28. Springer-Verlag, 1998.

27. A. Perrig and D. Song. Looking for diamonds in the desert (extending automatic protocol generation to three-party authentication and key agreement protocols). In *Proceedings of CSFW'00*. IEEE Computer Society Press, 2000.

28. D. Song, S. Berezin, and A. Perrig. Athena: a novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9:47–74, 2001.

29. S. D. Stoller. A bound on attacks on authentication protocols. In R. Baeza-Yates, U. Montanari, and N. Santoro, editors, *Proc. 2nd IFIP International Conference on Theoretical Computer Science: Foundations of Information Technology in the Era of Network and Mobile Computing*, pages 588–600. Kluwer, 2002.

30. J. Zhou. Further Analysis of the Internet Key Exchange Protocol. *Computer Communications*, 23(17):1606–1612, 2000.

1.	$A \to B:$	A, N_A
2.	$B \to S:$	$B, \{A, N_A, N_B\}_{K_{BS}}$
3.	$S \to A:$	$\{B, K_{AB}, N_A, N_B\}_{K_{AS}}$, $\{A, K_{AB}\}_{K_{BS}}$
4.	$A \to B:$	$\{A, K_{AB}\}_{K_{BS}}$, $\{N_B\}_{K_{AB}}$

Violated_goal: B authenticate S on KAB
Time: 0.02 seconds
Attack_trace:

1.	$i \to b:$	i, NA
2.	$b \to i(s):$	$b, \{i, NA, nb\}_{kbs}$
2.′	$i(b) \to s:$	$b, \{i, NA, nb\}_{kbs}$
3.	$s \to i:$	$\{b, kab, NA, nb\}_{kis}$, $\{i, kab\}_{kbs}$
4.	$i \to b:$	$\{i, NA, nb\}_{kbs}$, $\{nb\}_{\langle NA,nb\rangle}$

Figure 8: The Yahalom protocol and the new attack detected by the OFMC tool.

A The Yahalom Protocol: Constraints and their Reduction, and a New Attack

Figure 8 shows, on the left, the Yahalom protocol, which aims at distributing a session key K_{AB} to agents playing in the roles A and B with the help of a trusted server playing in the role S. The goal of the Yahalom protocol can be expressed as authentication (or agreement [20]) between A, B, and S. In particular, the intruder should not be able to replay or manipulate the session key K_{AB}. OFMC has discovered a new attack on the protocol violating this goal [5]. The right part of Figure 8 shows the attack trace output by OFMC.

In this trace, the intruder (under his real name i) plays in the role A, and the agents b and s play in the roles B and S. Upper-case letters denote variables and lower-case letters denote constants, e.g. NA is an arbitrary message (generated by the intruder) and nb is a fresh constant that agent b uses for the protocol variable NB. The notation $i(s)$ denotes that the intruder i poses as s.

The attack is as follows. According to the protocol, the intruder receives the new session key kab from s in step 3, along with a message encrypted for b, which should be forwarded to b in step 4. However, i replays the encrypted part of message 2 instead. This is accepted by b since, as expected, the message is encrypted with the key kbs and starts with the agent name i. Hence, b accepts the pair of nonces $\langle NA,nb\rangle$ as the session key issued by the server. Although the intruder doesn't "get in" with this attack (e.g. he did not make the agent b believe he talks with somebody else or find out a secret of other sessions), this state violates the authentication goal that any agent playing in the role B can rely on the integrity of the session-key: the intruder can make the agent b accept a fake key that was not originated by the server. A part of that key, i's nonce NA, is completely determined by the intruder.[8]

We will use this attack to illustrate the use of both kinds of constraints and constraint reduction. Let us assume that the initial intruder knowledge IK_0 contains only the names of the other agents and the key kis that the intruder shares with the server. In the attack, the intruder, playing role A, follows the protocol up to step 4 (he intercepts message 2 from b to s but then forwards it to s). Step 4 is where the attack actually takes place and we will focus on the constraints related to

[8] In [25] Paulson proved the security of the Yahalom protocol (including the goal we have found to be violated), but he used a typed model and the above attack exploits a type-confusion (between a key and a pair of nonces). The attack trace given above is similar to the attack originally described in [10]. There, the intruder listens to the communication between honest agents, and then, similar to our attack, tries to generate message 4 abusing the same confusion with the encrypted part of message 2 as in our attack (which is, however, impossible unless the intruder can guess the nonce N_B). Finally, note also that the attack we have detected is different from the one of [27], in which the intruder only makes the agent playing in the role B accept for the second time the key K_{AB} generated by the server. This is a replay attack. In our attack, the intruder makes the agent playing in the role B accept a key *different* from the one issued by the server.

this step.

Up to step 4, the only constraint is $\mathit{from}(NA, IK_0)$ since NA is the value initially generated by the intruder as a nonce. After intercepting message 2 from b to s, i's knowledge is simply $IK_1 = IK_0 \cup \{\!|i, NA, nb|\!\}_{kbs}$ as he cannot decrypt this message. After receiving message 3, the knowledge of i is $IK_2 = IK_1 \cup kab \cup nb \cup \{\!|i, kab|\!\}_{kbs}$ because i cannot decrypt the messages encrypted with kbs and the other messages contain only kab and nb as new information.

In step 4, the intruder sends $\langle \{\!|i, KAB|\!\}_{kbs}, \{\!|nb|\!\}_{KAB} \rangle$ to b and stores the constraint

$$\mathit{from}(\langle \{\!|i, KAB|\!\}_{kbs}, \{\!|nb|\!\}_{KAB} \rangle, IK_2) \tag{1}$$

containing the variable message part KAB. Note that, for clarity, we explicitly use here the pairing operator $\langle \cdot , \cdot \rangle$. Agent b will accept this message since the only message parts for which he expects a particular value are i and kbs, as well as nb, which b generated himself when he sent the second message of the protocol.

We can use this example also to illustrate D-from constraints and their reduction according to the rules of *D-Red* (the reduction of the from constraint (1) is similar to the reduction below, using *Red* instead of *D-Red*). Let us assume that we have the additional restriction that the messages learned in the previous step, i.e. the new intruder knowledge $IK_2 \setminus IK_1$, must be used in the generation of the message, and thus have the D-from constraint

$$\mathit{D\text{-}from}(\langle \{\!|i, KAB|\!\}_{kbs}, \{\!|nb|\!\}_{KAB} \rangle, IK_1, IK_2 \setminus IK_1) \ .$$

The first step of the reduction is an application of the rule G^{LD}_{pair}, which decomposes the pair and generates

$$\mathit{D\text{-}from}(\{\!|i, KAB|\!\}_{kbs} \cup \{\!|nb|\!\}_{KAB}, IK_1, IK_2 \setminus IK_1) \ .$$

There are three possibilities for the generation of the term $\{\!|i, KAB|\!\}_{kbs}$. First, the intruder could generate all parts of this message from his knowledge; however, this is impossible as the intruder does not know kbs. Second, the intruder can use the message obtained from the server s, i.e. message 3, so that $KAB = kab$ (and this possibility is not excluded by the D-from constraint, as the key kab is part of $IK_2 \setminus IK_1$). This reduction corresponds to a legal step of the protocol, as the agent in role A should indeed forward the message from the server. The third alternative is the most interesting one and we will focus on it: the intruder can replay the part of message 2 that is also encrypted with the key kbs. We can thus apply the rule G^{LD}_{unif1}, leading to the unification of the messages $\{\!|i, KAB|\!\}_{kbs}$ and $\{\!|i, NA, nb|\!\}_{kbs}$, i.e. we have $KAB = \langle NA, nb \rangle$ and the new constraint

$$\mathit{D\text{-}from}(\{\!|nb|\!\}_{\langle NA,nb \rangle}, IK_1, IK_2 \setminus IK_1) \ .$$

$\{\!|nb|\!\}_{\langle NA,nb \rangle}$ does not unify with any message in either the intruder knowledge or in the new intruder knowledge. We thus apply the G^{LD}_{scrypt} and G^{LD}_{pair} rules, which yield the constraint

$$\mathit{D\text{-}from}(NA \cup nb, IK_1, IK_2 \setminus IK_1) \ .$$

The nonce nb can now be trivially unified with the $nb \in IK_2 \setminus IK_1$, thereby using a term in the *NIK*-part, resulting in the from constraint

$$\mathit{from}(NA, IK_2) \ ,$$

so that we have the constraint set

$$\{\mathit{from}(NA, IK_0),\ \mathit{from}(NA, IK_2)\}.$$

This constraint set is simple as NA is a variable. Therefore the constraint reduction terminates here (as it is "lazy") and the attack trace contains this variable to denote that the particular choice of this message part is not important for the attack to work.

B Correctness and Completeness of *D-Red*

Theorem 2 *Let C be a well-formed constraint set. D-Red(C) is finite and $\vdash^D$ is well-founded. Moreover, $[\![\text{D-Red}(C)]\!] \subseteq [\![\lceil C \rceil]\!]$.*

Proof. The proof idea is to show that every *D-from* rule of *D-Red* is, in certain sense, a "refinement" of a *from* rule of *Red*, so that *D-Red* cannot allow more derivations than *Red*. Since Theorem 1 tells us that *Red* is correct, has finitely many solutions, and has no infinite derivation chains, we can conclude the same for *D-Red*.

More specifically, we first show that if $(C, \sigma) \vdash_r^D (C', \sigma')$ using a rule r of *D-Red*, then there is a corresponding rule $\overline{r}$ of *Red* such that $(\lceil C \rceil, \sigma) \vdash_{\overline{r}} (\lceil C' \rceil, \sigma')$. In other words, the rule r refines the rule $\overline{r}$. In particular, we have that G^{LD}_{pair} refines G^{L}_{pair}, G^{LD}_{scrypt} refines G^{L}_{scrypt}, both G^{LD}_{unif1} and G^{LD}_{unif2} refine G^{L}_{unif} and so forth.

All the cases are dealt with in a similar way and we focus here on a representative case and show that G^{LD}_{unif2} refines G^{L}_{unif}. Let C be a well-formed D-from constraint set such that G^{LD}_{unif2} is applicable to C. Hence there are $t, t', \tau, IK, NIK, C_1$ such that

$$C = \textit{D-from}(t \cup T, IK, t' \cup NIK) \cup C_1 \text{ and } \tau = mgu(t, t') \,.$$

The constraint set after application of the rule G^{LD}_{unif2} is $C' = (\textit{from}(T, t' \cup IK \cup NIK) \cup C_1)\tau$. As a consequence,

$$\lceil C \rceil = \textit{from}(t \cup T, t' \cup IK \cup NIK) \cup \lceil C_1 \rceil \text{ and } \lceil C' \rceil = (\textit{from}(T, t' \cup IK \cup NIK) \cup \lceil C_1 \rceil)\tau \,.$$

Since $t\tau = t'\tau$, the rule G^{L}_{unif} is applicable to $(\lceil C \rceil, \sigma)$ (for any substitution σ) and yields the result $(\lceil C' \rceil, \sigma\tau)$, which is what we wanted to show.

Since all derivations of *D-Red*(C) are possible in *Red*$(\lceil C \rceil)$, it follows that

$$[\![\lceil \textit{D-Red}(C) \rceil]\!] \subseteq [\![\textit{Red}(\lceil C \rceil)]\!] \,.$$

The correctness then follows by Theorem 1 and the fact that $[\![C]\!] \subseteq [\![\lceil C \rceil]\!]$, i.e.

$$[\![\textit{D-Red}(C)]\!] \subseteq [\![\lceil \textit{D-Red}(C) \rceil]\!] \subseteq [\![\textit{Red}(\lceil C \rceil)]\!] = [\![\lceil C \rceil]\!] \,.$$

We can directly use Theorem 1 to similarly show that *D-Red*(C) is finite and that $\vdash^D$ is well-founded. The main ideas here are that during the reduction only finitely many unifications are possible (as the number of variables is finite), and that between two unifications there can only be finitely many applications of other rules. □

Theorem 3 $[\![C]\!] \subseteq [\![\text{D-Red}(C)]\!]$ *for a well-formed constraint set C.*

Proof. The proof idea is to show that given a well-formed constraint set C and a solution $\sigma \in [\![C]\!]$, then there is at least one applicable constraint reduction rule such that the resulting constraint set still supports the solution σ (in a formal sense defined below).

Theorem 2 tells us that no infinite derivation chain is possible in *D-Red*, so the reduction must eventually terminate with a set of simple constraint sets (this is because we assumed $\sigma \in [\![C]\!]$ and hence *D-Red*(C) cannot be empty).

The main difficulty of this proof is the fact that the completeness requires that *D-Red* is performed on a well-formed constraint set (cf. Definition 2), but during the *D-Red* procedure the property (1) of the well-formedness can be destroyed by the analysis rules: an analysis rule *(a)* introduces a new constraint for the derivation of a key, where the intruder knowledge no longer contains the decryption key (hence, the intruder knowledge may be smaller than the intruder knowledge of all previous constraints), and *(b)* it adds the analyzed term to the intruder knowledge of the constraint to which it was applied (hence, the intruder knowledge may be larger than the intruder knowledge in all successive constraints). Note that here and below we speak of *previous* and *successive* constraints according to the order on the constraint set that is induced by its initial well-formedness.

Problem *(b)* can be easily overcome since to restore property (1) we perform the same analysis steps also on the successive constraints: these must allow the same derivations in the intruder knowledge as the constraint set the reduction started with was well-formed.

To tackle problem *(a)*, we relax the invariant in the proof: at any step during the proof we want to preserve the invariant that the constraint set is well-formed if one removes all constraints that were introduced by an analysis rule. For simplicity, we will still refer to this invariant as well-formedness. For the resulting simple constraints, we can restore the well-formedness in the original sense by simply deleting the constraints that were introduced by the analysis.[9]

More in detail, for every well-formed constraint set C, initial substitution σ, and solution $\tau \in [\![(C,\sigma)]\!]$, we show that, unless C is simple, there is a rule r of $\vdash^D$ such that $(C,\sigma) \vdash^D_r (C',\sigma')$ and $\tau \in [\![(C',\sigma')]\!]$. We then say that the constraint set after the rule application still *supports* the solution τ.

To do so, we first introduce the notion of a derivation tree. Since $\tau \in [\![C]\!]$, then $T\tau \subseteq \mathcal{DY}(IK\tau)$ for every constraint $\mathit{from}(T, IK) \in C$, and $T\tau \subseteq \mathcal{DY}((IK \cup NIK)\tau)$ and $T\tau \not\subseteq \mathcal{DY}(IK\tau)$ for every $D\text{-}\mathit{from}(T, IK, NIK) \in C$. We make explicit these derivations in the constraint set C by labeling every term in T with a particular derivation tree: A *$\mathcal{DY}$-derivation tree* (or simply *derivation tree* for short) is a binary tree, where leaves are messages and each node is an application of one of the $\mathcal{DY}$ rules. Every leaf and every node of the tree stands for a message, composed or decomposed from the respective subtrees. Hence, if $m \in \mathcal{DY}(IK)$, then there is a derivation tree such that m is the derived message at the root node and all leaves are in IK.

We mark the nodes of the tree with respect to NIK. A leaf node is marked iff it represents a message in NIK, and an inner node is marked iff one of its child nodes is marked. (A marked root node acts as a guarantee that some message from NIK was used.) We then say that a tree is *marked for use of NIK* iff all of its nodes are marked.

We say that a constraint set C is *labeled with $\mathcal{DY}$-derivation trees for a solution $\tau \in [\![C]\!]$* if in every constraint $\mathit{from}(T, IK) \in C$, every term $t \in T$ is labeled with the $\mathcal{DY}$-derivation tree of $t\tau$ from terms in $IK\tau$. Similarly, in every constraint $D\text{-}\mathit{from}(T, IK, NIK)$ every term $t \in T$ is labeled with the $\mathcal{DY}$-derivation tree of $t\tau$ from terms in $(IK \cup NIK)\tau$, the tree is marked for use of $NIK\tau$, and at least one term in T is labeled with a derivation tree that has a marked root node. To illustrate the proof, in Example 1 below we give a derivation using lazy reduction rules and the $\mathcal{DY}$-derivation trees.

Let a well-formed, non-simple constraint set C and a solution $\tau \in [\![C]\!]$ be given, where C is labeled with $\mathcal{DY}$-derivation trees for the solution τ. According to the order of the well-formed constraints, we pick the first constraint $c \in C$ that contains a message in its T-part that is not a variable. c can either be a from constraint or a D-from constraint. Observe that $[\![c_1]\!] = [\![c_2]\!]$ for $c_1 = \mathit{from}(T, IK)$ and $c_2 = D\text{-}\mathit{from}(T, \emptyset, IK)$, and that the derivations that can be considered

[9] This does not change the semantics of the constraint set. The constraint set is simple, therefore all constraints have only variables in their T-parts. Moreover, it is well-formed without the constraints C that were introduced by the analysis, so the constraints C can only have variables in their T-parts that were introduced by previous constraints. As these previous constraints have a smaller intruder knowledge, they are more restrictive, hence C is already entailed by them.

when applying *D-Red* to c_2 include the derivations that can be considered when applying *Red* on c_1. Therefore, we can here focus on the from constraints only. (In fact, the completeness proof of Theorem 1 given in [5] is a special case of this proof.)

Let us thus consider the case that the order induced by the well-formedness of C yields that $c = \mathit{D\text{-}from}(T, IK, NIK)$ is the first constraint in C that contains a term t in the T-part that is not a variable. We show that, depending on the root node n of the $\mathcal{DY}$-tree labeling t, we can find a reduction rule that is applicable and such that the resulting constraint set C' can again be labeled with $\mathcal{DY}$-trees according to τ (and hence the result still supports τ). We distinguish cases depending on the name of the $\mathcal{DY}$-rule labeling n and whether n is marked or not.

G_{axiom}/unmarked: This means that $t\tau \in IK\tau$. So, t can be unified with a term $t' \in IK$ and the G^{LD}_{unif1} rule is applicable to C, since t is not a variable. The unifier $\sigma = mgu(t, t')$ is compatible with τ. Hence, the resulting C' supports τ as all remaining terms can be labeled with the same trees as in C. Also, since the node is unmarked, there must be another term in the T-part of this constraint that is labeled with a tree with a marked root node.

G_{axiom}/marked: Like in the previous case, t can be unified under τ with a term $s \in NIK$, the G^{LD}_{unif2} rule is applicable to C, and the resulting C' supports τ: According to the rule, the D-from constraint is replaced with a respective from constraint and so all the terms of the T-part of the from constraint can be labeled with the same derivation trees because we have "used" some of the new knowledge. (Note that it is not a problem if none of these trees have a marked root node.)

G_{scrypt}/unmarked: Since t is not a variable, it must have the form $t = \{\!|t_1|\!\}_{t_2}$ for some terms t_1 and t_2. As a result, the rule G^{LD}_{scrypt} can be applied. The resulting constraint contains the terms t_1 and t_2, which can be labeled with the respective subtrees of the derivation tree of t. Hence, C' still supports τ. The marked case for G_{scrypt} is identical; note that then t_1 or t_2 are marked, hence there is still at least one marked term in the T-part.

The cases for G_{pair} are analogous to the cases for G_{scrypt}, mutatis mutandis.

For all analysis cases there are three problems that we have to tackle. (i) First, the intruder knowledge may contain variables, to which we cannot apply analysis rules as long as they are not instantiated. To see that the analysis of such a variable is not necessary during a derivation, we use the well-formedness of the constraints: every variable in the intruder knowledge must have been introduced in the T part of a previous constraint C_1 (which is already simple as we proceed with the reductions in the order induced by the well-formedness), and therefore represents a message the intruder has generated from his previous knowledge. Hence, the variable is not important for any derivation. Formally, let *tr* be the derivation tree the variable is labeled with when it is introduced in the T parts of the constraints in C_1. We can then replace with *tr* every occurrence of a leaf-node representing the value of the variable under τ. All transformed derivation trees have then still the same meaning, and the derivation no longer relies on the analysis of the variable.

(ii) The second problem stems from the fact that when analyzing an encrypted term $\{\!|m_1|\!\}_{m_2}$, then the constraints require that in the derivation of m_2 one does not need to use the term $\{\!|m_1|\!\}_{m_2}$ itself. There are examples where this seems like a restriction, namely if m_2 can only obtained by a different message m_3 in the intruder knowledge that is encrypted with the message $\{\!|m_1|\!\}_{m_2}$ as a symmetric key. However, this means only that before we can decrypt $\{\!|m_1|\!\}_{m_2}$ we must first analyze m_3 to obtain m_2. Therefore, when the root node of a derivation tree is an analysis node and some of the subtrees contain further analysis nodes, then we proceed with one of the "innermost" analysis nodes, i.e. such that no subtree contains analysis nodes.

(iii) The third problem is that an analysis rule may not be applicable if the terms obtained by the analysis are already contained in the IK-part or in the NIK-part of the constraint. (This is in fact an inefficiency that can be excluded by a more sophisticated analysis procedure.) In this case, we can replace in the $\mathcal{DY}$-trees every occurrence of the respective analysis subtree with a leaf, and proceed by looking for another applicable rule. Note that this cannot destroy the correct labeling. In particular this preserves the property that at least one derivation tree of a term in T is marked. To see this preservation, suppose the analysis node before the transformation is marked and the message obtained by the analysis is in IK, so after the transformation the respective node is unmarked. Then there is a derivation of this message possible without using NIK. Since $\tau \in [\![C]\!]$, there must be some other term t' of the T-part that can only be derived using NIK. Hence, t' must be labeled with a marked derivation tree as well.

A_{pair}/unmarked: as we remarked above, we consider an analysis node in the derivation tree that contains no further analysis nodes in the subtrees. For a A_{pair} node this means that the only subtree, i.e. the composed message to analyze, is a leaf (if it is a composition, then the analyzed parts are already contained in IK). Let m_1 be the ground message obtained by the analysis, and m be the analyzed term, hence $m = \langle m_1,m_2\rangle$ or $m = \langle m_2,m_1\rangle$ for some message m_2. Then, since m is a leaf, there is a term $m' \in IK$ such that $m'\tau = m$. By (i), it is not a restriction to assume that m' is not a variable but has the form $\langle m'_1,m'_2\rangle$ for some messages m'_1 and m'_2. It holds that $m'_1\tau = m_1$ or $m'_2\tau = m_2$, and, as argued in (iii), we can then exclude that $m_1 \in IK$. Therefore, the rule A^{LD}_{pair1} can be applied. All occurrences of m_1 in any derivation tree are replaced with a leaf node, since m'_1 (or m'_2) is now in IK. The labeling is still correct since m_1 in the derivation tree is unmarked.

A_{pair}/marked: m, m_1, and m_2 are the same as in the unmarked case, except that at least m and m_1 are marked. Hence, there is a message $m' \in NIK$ with $m' = \langle m'_1,m'_2\rangle$ and $m'\tau = m$. By (i) and (iii), we can exclude that $m' \in \mathcal{V}$ and that $m'_1 \in IK \cup NIK$. Then the rule A^{LD}_{pair2} is applicable and m'_1 is added to NIK. Replacing the marked analysis node with a marked leaf node for m_1 in the derivation tree yields a correct labeling of the resulting constraint.

A_{scrypt}/unmarked: similar to the previous cases, we consider a node of a derivation tree where a ground message m_1 is obtained by decrypting a message $m = \{\!|m_1|\!\}_{m_2}$ in $IK\tau$ for some message m_2, such that the derivation subtrees for $\{\!|m_1|\!\}_{m_2}$ and m_2 contain no analysis nodes. Therefore, the subtree for $\{\!|m_1|\!\}_{m_2}$ must be a leaf node (otherwise it can only be composed from messages in the intruder knowledge, which would imply that the intruder already knows m_2). So, there is a term $m' \in IK$ with $m'\tau = m$. By (i), we can assume that m' is not a variable, so that $m' = \{\!|m'_1|\!\}_{m'_2}$ for some messages m'_1 and m'_2 with $m'_1\tau = m_1$, and by (iii) we can assume that $m'_1 \notin IK$. Hence, we can apply the rule A^{LD}_{scrypt1}, which adds the analyzed term m'_1 to the intruder knowledge and the new constraint $\mathit{from}(m'_2, IK \setminus \{\!|m'_1|\!\}_{m'_2})$ to the constraint set. Now all occurrences of the analyzed term m_1 in the $\mathcal{DY}$-tree are replaced with a leaf-node; this labeling is correct, as $m'_1\tau = m_1$. In the newly added constraint $\mathit{from}(m'_2, IK \setminus \{\!|m'_1|\!\}_{m'_2})$, the term m'_2 is labeled with the subtree for m_2 of the derivation tree. This labeling is correct since this subtree cannot contain any further analysis operations; therefore it can contain only subterms of m_2 (if m_2 is non-atomic) that are present in $IK\tau$. In particular it cannot contain $\{\!|m_1|\!\}_{m_2}$. Hence, C' still supports τ.

A_{scrypt}/marked: Let m, m_1, and m_2 be like in the previous case but let m_1 be marked, so that also m is marked or m_2 is marked. Since m must be again a leaf node, there is a term m' in IK or in NIK, depending on whether m is unmarked or marked, such that $m'\tau = m$. By (i), we can again exclude the case that m' is a variable, and thus there are m'_1 and m'_2 like in the previous case. By (iii), we can exclude that m'_1 is in IK. In the case that m is unmarked (and hence $m' \in IK$ and m_2 marked), we can apply A^{LD}_{scrypt2}. Again, all occurrences of the analyzed term m'_1 in the derivation trees are replaced with a leaf node, as $m'_1\tau = m_1$. As m_2 is marked, we can correctly label m'_2 in the new from constraint with the derivation tree of m_2: in the derivation of m'_2 we exploit knowledge from NIK and, as this derivation does not contain any further analysis, the term m' is not needed.

As in the last case, we consider the case that m_1 is marked (so m_2 is not necessarily marked). In this case, it holds that $m_1' \in NIK$ and by (iii) we can again exclude that $m' \in IK \cup NIK$, so the rule A^{LD}_{scrypt3} is applicable. In the resulting constraint, the labelings of the T-part are again transformed by replacing all occurrences of m with a leaf node. Also in the new from constraint, the term m_2' is labeled with the derivation tree of m_2 which contains no further analysis rules, hence in this derivation m is never necessary and therefore the labeling is correct.

We have thus considered all possible cases and shown that we can always perform at least one derivation step which keeps the invariants of well-formedness (with the explained restriction) and the correct labeling. Hence, *D-Red* is complete. □

Example 1. As a simple example, consider $C = \{\mathit{from}(\langle K,m\rangle, \{\!|m|\!\}_k \cup \{\!|k|\!\}_{\{\!|m|\!\}_k})\}$, which has, among others, the solution $\tau = [K \mapsto \{\!|m|\!\}_k]$, where K is a variable while the other identifiers are constants. We can label the message $\langle K,m\rangle$ with the derivation tree for the message $\langle K,m\rangle\tau = \langle \{\!|m|\!\}_k, m\rangle$ as follows:

$$\dfrac{\dfrac{}{\{\!|m|\!\}_k}\,G_{\text{axiom}} \qquad \dfrac{\dfrac{\dfrac{}{\{\!|m|\!\}_k}\,G_{\text{axiom}} \quad \dfrac{}{\{\!|k|\!\}_{\{\!|m|\!\}_k}}\,G_{\text{axiom}}}{k}\,A_{\text{scrypt}} \quad \dfrac{}{\{\!|m|\!\}_k}\,G_{\text{axiom}}}{m}\,A_{\text{scrypt}}}{\langle \{\!|m|\!\}_k, m\rangle}\,G_{\text{pair}}$$
$$\boxed{\langle K,m\rangle}$$

where $IK\tau = \{\!|m|\!\}_k \cup \{\!|k|\!\}_{\{\!|m|\!\}_k}$.

We now show how the reduction proceeds according to the previous case split. First, the root of the only term to generate is G_{pair}. Therefore we can apply the G^L_{pair} to obtain $\mathit{from}(K \cup m, \{\!|m|\!\}_k \cup \{\!|k|\!\}_{\{\!|m|\!\}_k})$ labeled as follows (recall that we consider the solution $\tau = [K \mapsto \{\!|m|\!\}_k]$ and $IK\tau = \{\!|m|\!\}_k \cup \{\!|k|\!\}_{\{\!|m|\!\}_k}$):

$$\dfrac{}{\{\!|m|\!\}_k}\,G_{\text{axiom}} \qquad\qquad \dfrac{\dfrac{\dfrac{}{\{\!|m|\!\}_k}\,G_{\text{axiom}} \quad \dfrac{}{\{\!|k|\!\}_{\{\!|m|\!\}_k}}\,G_{\text{axiom}}}{k}\,A_{\text{scrypt}} \quad \dfrac{}{\{\!|m|\!\}_k}\,G_{\text{axiom}}}{m}\,A_{\text{scrypt}}$$
$$\boxed{K} \qquad\qquad\qquad\qquad \boxed{m}$$

Since K is a variable in the constraint set, we can only proceed by deriving m. As the root of the derivation tree is an analysis operation and one subtree contains a further analysis step, we proceed with this innermost analysis. The respective analysis rule decrypts $\{\!|k|\!\}_{\{\!|m|\!\}_k}$, adds k to the intruder knowledge and the new constraint that the key term, $\{\!|m|\!\}_k$, can be derived from the rest of the knowledge, i.e. $\mathit{from}(\{\!|m|\!\}_k, \{\!|m|\!\}_k), \mathit{from}(K \cup m, k \cup \{\!|m|\!\}_k \cup \{\!|k|\!\}_{\{\!|m|\!\}_k})$, with the labeling:

$$\dfrac{}{\{\!|m|\!\}_k}\,G_{\text{axiom}} \qquad \dfrac{}{\{\!|m|\!\}_k}\,G_{\text{axiom}} \qquad \dfrac{\dfrac{}{k}\,G_{\text{axiom}} \quad \dfrac{}{\{\!|m|\!\}_k}\,G_{\text{axiom}}}{m}\,A_{\text{scrypt}}$$
$$\boxed{\{\!|m|\!\}_k} \qquad\qquad \boxed{K} \qquad\qquad \boxed{m}$$

The first constraint is easily handled by the G^L_{unif} rule; we proceed then with the analysis of $\{\!|m|\!\}_k$, as the $\mathcal{DY}$-tree contains no further analysis operations. This introduces the new constraint for the derivation of k, i.e. $\mathit{from}(k, k \cup \{\!|k|\!\}_{\{\!|m|\!\}_k}), \mathit{from}(K \cup m, m \cup k \cup \{\!|m|\!\}_k \cup \{\!|k|\!\}_{\{\!|m|\!\}_k})$, with the labeling:

$$\dfrac{}{k}\ G_{\text{axiom}} \qquad \dfrac{}{\{\!|m|\!\}_k}\ G_{\text{axiom}} \qquad \dfrac{}{m}\ G_{\text{axiom}}$$

$$\boxed{k} \qquad \boxed{K} \qquad \boxed{m}$$

Two further applications of G^L_{unif} then result into the simple constraint set that supports τ, i.e. $\mathit{from}(K, m \cup k \cup \{\!|m|\!\}_k \cup \{\!|k|\!\}_{\{\!|m|\!\}_k})$.

For simplicity, we have considered only a from constraint. It is not difficult to see that, for instance, the D-from constraint

$$\mathit{D\text{-}from}(\langle K,m\rangle, \{\!|m|\!\}_k, \{\!|k|\!\}_{\{\!|m|\!\}_k})$$

supports exactly the same solution σ, which can be found by applying the rules of *D-Red*.