Diffie-Hellman without Difficulty (Extended Version)*

Sebastian Mödersheim

DTU Informatics
samo@imm.dtu.dk

Abstract. An excellent way for a protocol to obtain shared keys is Diffie-Hellman. For the automated verification of security protocols, the use of Diffie-Hellman poses a certain amount of difficulty, because it requires algebraic reasoning. Several tools work in the free algebra and even for tools that do support Diffie-Hellman, the algebraic reasoning becomes a bottleneck.
We provide a new relative-soundness result: for a large class of protocols, significantly restricting the abilities of the intruder is without loss of attacks. We also show the soundness of a very restrictive encoding of Diffie-Hellman proposed by Millen and how to obtain a problem that can be answered in the free algebra without increasing its size upon encoding. This enables the efficient use of free-algebra verification tools for Diffie-Hellman based protocols and significantly reduces search-spaces for tools that do support algebraic reasoning.

1 Introduction

Many modern security protocols like IKE/IPSec [11] employ the Diffie-Hellman key exchange [9] to obtain a shared key between two parties. The reason that Diffie-Hellman is so popular is that it is a simple mechanism with excellent properties. The main problem for the verification of Diffie-Hellman based protocols is that they rely on an algebraic property of modular exponentiation (we omit the modulus in our notation): $\mathsf{exp}(\mathsf{exp}(B,X),Y) \approx \mathsf{exp}(\mathsf{exp}(B,Y),X)$. Interpreting the message terms of a protocol in a free algebra (ignoring said property) gives a nonsensical model where agents can never arrive at a shared key.

However, a number of successful protocol verification methods do not support algebraic reasoning at all, for instance ProVerif [5], SATMC [3], and Scyther [7]. Also for tools that do support algebraic reasoning like OFMC [4], CL-AtSe [22], MaudeNPA [10], and an extension of ProVerif [13], the algebraic reasoning means an extra burden, in particular as it affects the most basic components of a verification tool, namely unification and intruder deduction.

This paper is based on the observation that much of the algebraic reasoning related to Diffie-Hellman is actually not very "interesting". Consider the situation that an honest agent a wants to start an exchange with the dishonest

* The author thanks Luca Viganò and the anonymous reviewers for helpful comments.

intruder i.[1] Then a first generates a secret x and sends the half-key $\mathsf{exp}(g, x)$. The intruder can now choose a half-key of his own, but it does not necessarily have the form $\mathsf{exp}(g, \cdot)$ because a will not be able to check that. The intruder could thus choose any term t, and the resulting full-key will be $\mathsf{exp}(t, x)$. Not knowing x, the intruder can obtain this full-key only for certain choices of t. The "well-typed" choice (as the protocol intends it) is $t = \mathsf{exp}(g, y)$ for a value y that he knows. However there is also an infinite number of "ill-typed" choices: $t = g$, $t = \mathsf{exp}(\mathsf{exp}(g, y_1), y_2)$, etc. The intuition is now that the ill-typed choices are just slight variations that are not particularly interesting, because that they do not enable attacks that are impossible using well-typed choices. This intuition is however not correct for all protocols in general. In a badly designed protocol, confusion may arise which messages or message parts are actually meant as Diffie-Hellman half-keys. Consider for instance the protocol:

$$\begin{aligned} A \to B &: \mathsf{sign}(\mathsf{inv}(\mathsf{pk}(a)), g) \\ A \to B &: \mathsf{sign}(\mathsf{inv}(\mathsf{pk}(a)), \mathsf{exp}(g, X)) \\ B \to A &: \mathsf{sign}(\mathsf{inv}(\mathsf{pk}(b)), \mathsf{exp}(g, Y)) \end{aligned}$$

Here, A acknowledges the group g in the first message. Since B will construct the full key as $\mathsf{exp}(t, Y)$ for whatever value he receives from A as a half-key in the second step, the intruder could replay the first message from A as the second message to achieve $t = g$ (i.e. an ill-typed solution) and thus know the key that B believes to have with A. The problem with this protocol is of course that the first and second message are similar enough to allow for a confusion. Our result will only apply to protocols that in some way exclude such confusions.

Contributions This paper establishes sufficient conditions for Diffie-Hellman based protocols under which we can restrict the intruder choices for all exponentiations to *well-typed* ones. This is a *relative soundness* result in the style of [12, 18, 15, 14, 2]: if a given protocol has an attack in the unrestricted model, then it also has an attack in the restricted model. It is thus without loss of generality to employ the restricted model in verification tools.

Further, with this result, we can justify a very restrictive model of Diffie-Hellman proposed by Millen [8, 17]. This model handles Diffie-Hellman more abstractly, reducing the amount of algebraic reasoning necessary and allows also for a free algebra encoding as we briefly sketch. Therefore these results are beneficial both to tools that do employ algebraic reasoning and those that do not.

Main Argument Like [2], we use as a main argument a constraint reduction technique [19, 21, 6, 4]. While this technique is normally used for *verification*, we use it in this paper as a *proof* technique. Roughly speaking, the technique consists of collecting constraints about what messages the intruder must be

[1] It is of course good practice in protocol verification to allow the intruder to play under his real name in any of the protocol roles (except trusted third parties) in order to model dishonest or compromised participants.

able to construct from what knowledge, and a sound and complete reduction procedure for these constraints. Since this technique avoids exploring the entire space of possible intruder messages and rather works in a demand driven way, we like to refer to this technique as the *lazy intruder*.

The use of the lazy intruder in this paper is based on the idea that every attack to a protocol is a solution of a well-formed lazy intruder constraint: it represents symbolically the requirements that the exchanged messages have to fulfill and thus an entire class of attacks, including possibly both well-typed and ill-typed ones (with respect to the Diffie-Hellman exponentiations). We show that the constraint reduction of the lazy intruder never makes an ill-typed choice and eventually arrives at *simple* form that supports at least one well-typed attack. From the completeness of the lazy intruder then follows that if there is an attack then there is also a well-typed one.

We note that while the lazy intruder is normally used as a verification technique for a bounded number of session, our result is neither a verification technique itself nor is it limited to a bounded number of sessions. Also, our modifications of the lazy intruder technique for Diffie-Hellman are orthogonal to those of [6] which aim for a verification technique.

Plan The rest of this paper is organized as follows. § 2 introduces the message and intruder model. § 3 reviews the lazy intruder technique for the free algebra. § 4 adapts the lazy intruder to Diffie-Hellman and proves the main completeness result. § 5 uses our result to justify Millen's restricted model and briefly discusses how to encode things into a free algebra model. In § 6 we conclude with a discussion of the related work.

2 Preliminaries

2.1 Messages

Following the line of black-box cryptography models, we employ a term algebra to model the messages that participants exchange. Let Σ be countable signature and $\mathcal{V}$ be a countable set of variable symbols disjoint from Σ. As a convention, constants and function symbols are denoted using identifiers that start with a lower-case letter and variables using upper-case letters. The signature is partitioned into the set Σ_0 of constants, the set of Σ_p of ("public") operations, and the set Σ_m of ("private") mappings (explained below). We use standard notions about terms such as *ground* (without variables), *subterm* (denoted $s \sqsubseteq t$), substitutions (denoted with σ, τ), *set of most general unifiers* (denoted mgu).

The constants represent agents, keys, nonces, and the like. The function symbols of Σ_p represent operations on messages that every agent can perform. In this paper we use the following function symbols Σ_p:

– $\mathsf{crypt}(k, m)$ represents the asymmetric encryption of message m with public key k.

- $\mathsf{scrypt}(k, m)$ represents the symmetric encryption of message m with symmetric key k; we assume that this primitive includes also integrity protection such as a MAC.
- $[m_1, \ldots, m_n]_n$ (for every $n \geq 2$) representing the concatenation of n messages $m_1, \ldots, m_n$. We use this family of operators to abstract from the details of structuring messages in the implementation. This model of concatenation is helpful for handling of the context around Diffie-Hellman half-keys in $ 4.1: using a conventional nested binary concatenation operator instead, would make the argumentation significantly more complex.
- $\mathsf{sign}(k, m)$ represents the signature of message m with private key k.
- $\mathsf{exp}(B, X)$ represents a modular exponentiation where B is the basis, and X is the exponent (and we omit the modulus in the abstract term here).

Mappings The symbols of Σ_m represent mappings such as $\mathsf{inv}(k)$ that yields the private key of public key k and that is obviously not a "public" operation. Such mappings are also convenient to specify key infrastructures, e.g., $\mathsf{pk}(a)$ denoting the public key of agent a. When dealing with only ground terms, one may regard these mappings directly as operations on Σ_0 rather than as function symbols of the term algebra. However, as we are using symbolic terms like $\mathsf{pk}(A)$ (in a description that is parametrized over a variable A), we need to include these symbols in the term algebra. Terms like $\mathsf{inv}(\mathsf{pk}(A))$ that do not contain public function symbols of Σ_p will actually be regarded as "atomic" terms.

Algebraic Equations We interpret terms in the *quotient algebra* under the equation $\mathsf{exp}(\mathsf{exp}(\mathsf{B}, \mathsf{X}), \mathsf{Y}) \approx \mathsf{exp}(\mathsf{exp}(\mathsf{B}, \mathsf{Y}), \mathsf{X})$. Thus, two terms are interpreted as equal iff they are syntactically equal modulo application of this equation. This property of exponentiation is in a sense the "minimal" algebraic theory that is necessary for considering Diffie-Hellman-based protocols: without it, even the "legal" execution of the protocol by honest agents is impossible.

Intruder Deduction Informally, the intruder can compose new terms applying public functions of Σ_p to terms he knows, and he can decompose terms when he knows the necessary keys. The latter formalized by a function $ana(\cdot)$ that takes as argument a message m and returns a set of potential ways to extract information from m. Each way to extract information has the form (K, P) where P ("plaintexts") is a set of messages that can be extracted when the messages

K (“keys”) are known.[2] In this paper we use:

$$ana(m) = \begin{cases} \{(\{\mathsf{inv}(k)\}, \{p\})\} & m = \mathsf{crypt}(k,p) \\ \{(\{k\}, \{p\})\} & m = \mathsf{scrypt}(k,p) \\ \{(\emptyset, \{p_1,\dots,p_n\})\} & m = [p_1,\dots,p_n]_n \\ \{(\emptyset, \{p\})\} & m = \mathsf{sign}(\mathsf{inv}(k),p) \\ \emptyset & \text{otherwise} \end{cases}$$

Definition 1. *We denote with $M \rhd m$ that the intruder can derive the ground message m when knowing the set of ground messages M. We define $\rhd$ as the least relation that satisfies the following rules:*

(D) *$M \rhd m$ for all $m \in M$,*
(G) *if $M \rhd t_1, \dots, M \rhd t_n$, then also $M \rhd f(t_1,\dots,t_n)$ for all $f \in \Sigma_p^n$,*
(A) *if $M \rhd m$ and $(K,P) \in ana(m)$ and $M \rhd k$ for all $k \in K$, then also $M \rhd p$ for all $p \in P$.*

Here, all terms here are interpreted w.r.t. their algebraic properties without an explicit rule.

Example 1. Consider the knowledge $M = \{\mathsf{scrypt}(\mathsf{exp}(\mathsf{exp}(g,x),y),m), \mathsf{exp}(g,y), x\}$. Then it holds that $M \rhd m$, as can be represented by the following proof tree:

$$\dfrac{\dfrac{}{M \rhd \mathsf{scrypt}(\mathsf{exp}(\mathsf{exp}(g,x),y),m)}\,(D) \qquad \dfrac{\dfrac{}{M \rhd \mathsf{exp}(g,y)}\,(D) \quad \dfrac{}{M \rhd x}\,(D)}{M \rhd \mathsf{exp}(\mathsf{exp}(g,x),y)}\,(G)}{M \rhd m}\,(A)$$

While this definition is given only for ground m and M, we will in the next section use the symbol $\rhd$ in *constraints* that contain variables.

3 The Lazy Intruder—Revisited

We now review the constraint reduction technique of [19, 21, 6, 4] that we refer to as the lazy intruder, and that we will use as a convenient way in our argumentation. Throughout the paper we will take for granted that every trace, in particular every attack trace, can be represented by lazy intruder constraints, even if one may be interested in a completely different verification technique.

[2] For the purpose of automated analysis methods, it is usually necessary to require that for every $(K,P) \in ana(m)$, K and P are subterms of m modulo bounded application of mapping symbols (e.g. allowing $\mathsf{inv}(k)$ where k is subterm of m) to prevent looping during analysis; we do not need to make this restriction here, because non-termination is never a problem in our arguments.

$$\frac{\phi\tau}{\phi \wedge (\{s\} \cup M \rhd t)} \text{ Unify } (\tau \in \mathsf{mgu}(\{s,t\}), s,t \notin \mathcal{V}, \mathit{admitted}(s), \mathit{admitted}(t))$$

$$\frac{\phi \wedge (M \rhd t_1) \wedge \ldots \wedge (M \rhd t_n)}{\phi \wedge (M \rhd f(t_1, \ldots, t_n))} \text{ Generate}, f \in \Sigma_p \setminus \{\mathsf{exp}\}$$

$$\frac{\phi \wedge (\{s\} \cup P \cup M \rhd t) \wedge \bigwedge_{k \in K} \{s\} \cup M \rhd k}{\phi \wedge (\{s\} \cup M \rhd t)} \text{ Ana } ((K,P) \in \mathsf{ana}(s), s \notin \mathcal{V})$$

Fig. 1. The lazy intruder reduction rules.

Semantics of Constraints We consider *constraints* which are conjunctions of $M \rhd m$ statements where both M and m may contain variables. An *interpretation* $\mathcal{I}$ assigns a ground term to every variable; we write $\mathcal{I}(v)$ to denote the interpretation of a variable v and extend this notation to messages, sets of messages, and constraints as expected. We inductively define the relation $\mathcal{I} \models \phi$ to formalize that *interpretation* $\mathcal{I}$ *is a model of constraint* ϕ:

$$\begin{aligned} \mathcal{I} \models M \rhd m & \text{ iff } \mathcal{I}(M) \rhd \mathcal{I}(m) \\ \mathcal{I} \models \phi \wedge \psi & \text{ iff } \mathcal{I} \models \phi \text{ and } \mathcal{I} \models \psi \end{aligned}$$

A constraint is *satisfiable* if it has at least one model.

Constraint Reduction The core of the lazy intruder is a set of reduction rules based on which we can check in finitely many steps whether a given constraint is satisfiable. Before we discuss the rules shown in Fig. 1, let us first review the idea of constraint reduction in a conceptual way. The reduction rules work similar to the rules of a proof calculus in several regards. A rule of the form $\frac{\phi'}{\phi}$ tells us that, in order to show the satisfiability of constraint ϕ (the proof goal), it suffices to show the satisfiability of constraint ϕ' (the sub goal). So we apply the rules in a backward fashion in the search for a satisfiability proof. This process succeeds once we find a *simple* constraint which is one that consists only of conjuncts of the form $M \rhd v$ where v is a variable. A simple constraint is obviously satisfiable: the intruder can choose for each variable an arbitrary message that he can construct. In fact, the laziness of the intruder manifests itself exactly here in avoiding the exploration of choices that do not matter.

Comparing to a proof calculus, one could call the simple constraints the "axioms" and we check whether for a given constraint ϕ any proof can be constructed using the reduction rules that has ϕ as a root and only simple constraints ("axioms") as leaves. *Soundness* of such a calculus of reduction rules means that we never obtain a "proof" for an unsatisfiable constraint, and *completeness* means that every satisfiable constraint has a proof. There are further relevant properties such as finiteness of the set of reachable proof states, and the completeness of certain proof strategies. These play a minor role in this paper because we do not use the lazy intruder to implement an efficient model checker, but rather use the existence or non-existence of certain reduction as a proof argument in the proof of our main theorems.

Unify Let us now consider the details of the rules in Fig. 1. The Unify rule says that one way for the intruder to produce a term t is to use any term s in his knowledge that can be unified with t. Here, $\mathsf{mgu}(\{s,t\})$ means the set of most general unifiers between s and t (note that there can be several in unification modulo the property of exponentiation). In case τ is such a unifier, we have solved the constraint $\{s\} \cup M \rhd t$ and apply τ to the remaining constraint ϕ to be solved. We make here also an essential restriction: neither s nor t shall be variables. If t is a variable, then the constraint $\{s\} \cup M \rhd t$ is already simple and should not be reduced to achieve the laziness. The case that s is a variable is more involved. Roughly speaking, such a variable will represent a value chosen by the intruder "earlier" and so whatever it is, he can also generate the same value from M already. This will be made precise below with the notion of well-formed constraints and in the completeness proof. The *admitted*$(\cdot)$ side conditions of the rule are related to our way of handling the exponentiations; for now let us assume they are simply true for all terms.

Generate The Generate rule tells us that the intruder can generate the term $f(t_1, \ldots, t_n)$ if f is a public symbol of Σ_p and if the intruder can generate all the subterms $t_1, \ldots, t_n$. So this simply represents the intruder applying a public function (such as encryption) to a set of terms he already knows. We exclude here the exp symbol because we will treat exponentiation in a special way below.

Ana The Ana rule represents the intruder trying to decompose messages in his knowledge such as decrypting with known keys. Given the intruder knows a message m from which he can learn P provided he knows K, we can go to a new constraint where the knowledge is augmented with the messages of P and where we have the additional constraints that the intruder can generate every $k \in K$. In fact, in actual implementations this rule must be carefully implemented to avoid non-termination of the search. For the same reason as in the case of the Unify rule, we do not analyze s if it is a variable, because then–the way we use it–it represents a message created earlier by the intruder.

Example 2. Consider the constraint

$$\phi = M_0 \rhd [A, B, N]_3 \ \wedge \ M \rhd [\mathsf{scrypt}(k, [A, B, K']_3), \mathsf{scrypt}(K', N)]_2$$

where $M_0 = \{i, a, b, k\}$ and $M = M_0 \cup \{\mathsf{scrypt}(k, [A, B, N]_3)\}$. This constraint is satisfiable as the following reduction shows:

$$\cfrac{\cfrac{\cfrac{\cfrac{}{M_0 \rhd A \wedge M_0 \rhd B \wedge M_0 \rhd N \wedge M \rhd N}\ \text{(Simple Constraint)}}{M_0 \rhd A \wedge M_0 \rhd B \wedge M_0 \rhd N \wedge M \rhd \mathsf{scrypt}(N, N)}\ \text{Generate}}{M_0 \rhd A \wedge M_0 \rhd B \wedge M_0 \rhd N \wedge M \rhd \mathsf{scrypt}(k, [A, B, K']_3) \wedge M \rhd \mathsf{scrypt}(K', N)}\ \text{Unify}(K' = N)}{\phi}\ \text{Generate}^*$$

Lemma 1. *All rules of the calculus are sound.*

Proof. Unify *Rule:* Given an interpretation $\mathcal{I}_0$ such that $\mathcal{I}_0 \models \phi\tau$. Then there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \phi$ and $\mathcal{I}(v) = \mathcal{I}(v\tau)$ for every $v \in \mathsf{dom}(\tau)$. Moreover, $\mathcal{I}(s) = \mathcal{I}(t)$ and thus $\mathcal{I} \models \{s\} \cup M \rhd t$. Thus also $\phi \wedge (\{s\} \cup M \rhd t)$ is satisfiable.

Generate *Rule:* Straightforward.

Ana *Rule:* Given $\mathcal{I} \models \phi \wedge (\{s\} \cup P \cup M \rhd t) \wedge \bigwedge_{k \in K} \{s\} \cup M \models \rhd k$ and $(K, P) \in ana(s)$. Let $M_0 = \mathcal{I}\{s\} \cup M$; then $M_0 \rhd \mathcal{I}(k)$ for every $k \in K$, and since $\mathcal{I}(s) \in M_0$, then also $M_0 \rhd \mathcal{I}(p)$ for every $p \in P$. Since $\mathcal{I} \models \{s\} \cup P \cup M \rhd t$, we have thus also $\mathcal{I} \models \{s\} \cup M \rhd t$. $\square$

3.1 Well-Formedness

We can define an order on the conjuncts of constraints, talking about earlier/later constraints. This order is essential for the constraint reduction. The idea is that the intruder does a sequence of actions during an attack and his knowledge monotonically grows with every message he learns. Also variables that occur in messages sent by honest agents must have appeared in previous messages and thus represent values that depend on the choice of the intruder (though they might not be chosen by the intruder himself).

Definition 2. *We say a constraint ϕ is* well-formed *if it has the form (modulo reordering conjuncts)*

$$\phi = \bigwedge_{i=1}^{n} M_i \rhd t_i$$

such that for $i \leq j$, $M_i \subseteq M_j$—expressing that the intruder never forgets—and $\mathsf{vars}(M_i) \subseteq \bigcup_{j=1}^{i-1} \mathsf{vars}(t_j)$*—all variables arise from intruder choices.*

For the free algebra without exponentiation, the calculus is already complete on well-formed constraints [19]. We show such a completeness result in Theorem 1 for a modification of the lazy intruder that supports exponentiation in the way it is used by Diffie-Hellman-based protocols.

3.2 Symbolic Transition Systems

Throughout this paper, we always argue in terms of lazy intruder constraints, taking for granted that the behavior of a protocol can always be described in terms of symbolic transitions. In fact, as argued in [20], we can see all standard protocol description languages such as (multi)-set rewriting, process calculi, Horn clauses, or strand/bundles as giving rise to symbolic transition systems: the acceptable messages of honest agents can be characterized by terms with variables where the variables represent subterms for which the agent does not expect a concrete particular value; the behavior of the agent depends on such variables (the variables can occur in subsequent outgoing messages). The lazy intruder technique then represents executing steps in a transition system without ever instantiating variables with concrete values and just retaining constraints about them (that are then checked for satisfiability). So in principle the lazy intruder is compatible with many formalisms to describe the behavior of honest agents (although extensions need to be made when considering for instance negation of predicates). We do not discuss this here further and refer to [20]. Again we emphasize that our approach uses the lazy intruder only as a proof argument and the result can be applied for any kind of protocol verification technique.

4 Handling Diffie-Hellman Exponentiation

4.1 Context for Half-Keys

For our result, we need that the exchange of Diffie-Hellman half-keys $\mathsf{exp}(g, x)$ and $\mathsf{exp}(g, y)$ is clearly distinguished from other parts of the protocol. We do not prescribe how this is done, e.g., one may be using a unique tag. Formally, we require that all half-keys that are exchanged are embedded into a context $C[t_i] = f(t_1, \ldots, t_n)$ for some $1 \leq i \leq n$ and some $f \in \Sigma_p \setminus \{\mathsf{exp}\}$. For instance a context may be $C[\cdot] = [\mathsf{dh}, B, \cdot]_3$, i.e., a triple that identifies by the tag dh that this message is meant as a Diffie-Hellman key, that the intended recipient is B. Here, the variable B is a parameter of the context that will be instantiated in concrete protocol runs. For simplicity, we do not bother with instantiation of contexts, and suppose in the following that $C[t]$ is ground if t is; the extension to parametrized contexts is as expected. Our main requirement on contexts is that the context $C[v]$ for a fresh variable v cannot be unified with any other non-variable subterm of a message of the protocol. Intuitively, no message part that is meant for a different purpose can be mistaken as a Diffie-Hellman half-key.

4.2 An Extended Notion of Simplicity

In all lazy intruder approaches so far, the notion of simplicity is $M \rhd v$ for a variable v, while any other, non-variable, term is considered as not being simple. The key to a "very lazy" approach to Diffie-Hellman exponentiation is to use the following extended notion of simplicity.

Definition 3. *We define* simplicity *of a constraint ϕ, and write simple(ϕ), as the least relation satisfying:*

- *simple($M \rhd v$) if $v \in \mathcal{V}$,*
- *simple(ϕ_1) and simple(ϕ_2) implies simple($\phi_1 \wedge \phi_2$)*
- *simple(ϕ) implies simple($\phi \wedge (M_0 \rhd v) \wedge (M \rhd \mathsf{exp}(v, c))$), provided that v is a variable, c is a constant, $M_0 \subseteq M$, and $\mathsf{exp}(g, c) \in M$.*

The last closure property of the simplicity relation is actually the key idea: if the intruder has to generate an exponentiation $\mathsf{exp}(v, c)$ and v is a value that he can choose himself earlier (at knowing M_0), and he knows the public value $\mathsf{exp}(g, c)$ then he has infinitely many choices for v that work. (We require below that the intruder knows g.) The "well-typed" ones (in the sense that we pursue in this work) would be $v = \mathsf{exp}(g, z)$ for some nonce z that he chooses. He may also choose "ill-typed" ones like $v = g$ or $v = \mathsf{exp}(\mathsf{exp}(g, z_1), z_2)$ etc. Our definition of simplicity allows the intruder to be lazy at this point and not choose v; we thus broaden the cases where the intruder may stop working and thus make him lazier. Still, it is ensured that every simple constraint is satisfiable (i.e. there is at least one solution).

We also define the restriction on the admissible terms for the Unify rule:

Definition 4. *We say a* term t is admitted for reduction*, written admitted*(t)*, if the following two conditions hold:*

- *t is not a variable and*
- *if $t = \mathsf{exp}(t_1, t_2)$ then t_1 is neither a variable nor an exponentiation.*

Example 3. The constraint $M_0 \rhd \mathsf{scrypt}(k, GX) \wedge M \rhd \mathsf{scrypt}(\mathsf{exp}(GX, y), P)$ for $M_0 = \{g, k\}$ and $M = M_0 \cup \{\mathsf{scrypt}(k, \mathsf{exp}(g, y))\}$ can be reduced (using rule Generate) to $M_0 \rhd GX \wedge M \rhd \mathsf{exp}(GX, y) \wedge M \rhd P$. This constraint is still not simple, because the second conjunct is an exponentiation $\mathsf{exp}(GX, y)$ and $\mathsf{exp}(g, y) \notin M$. The Generate rule cannot be applied (because the top symbol is exp) and the Unify rule cannot be applied (because $\mathsf{exp}(GX, y)$ is not admissible and there is no unifiable term in M). However we can apply the Ana rule to decrypt $\mathsf{scrypt}(k, \mathsf{exp}(g, y))$; after resolving the new conjunct $M \rhd k$, the constraint is simple.

4.3 Well-formedness for Diffie-Hellman

So far, our model is completely untyped. In fact we will in the following label a subset of the variables and constants with the types xp (for exponent) and hk (for half-key). These reflect only the intentions of the protocol, without excluding ill-typed assignments.

Definition 5. *Consider that k is a subterm of t at position p. Then we say that p is a* symmetric encryption position*, if $p = p' \cdot 0$ and the subterm of t at position p' is $\mathsf{scrypt}(k, \cdot)$. We say that p is the root position of t if $p = \epsilon$.*

We say a constraint ϕ is well-formed for Diffie-Hellman *w.r.t. a half-key context $C[\cdot]$, if it is both well-formed in the sense of Definition 2, and every occurrence of* exp *and terms of type* xp*,* hk*, and the context $C[\cdot]$ is in one of the following forms:*

- *$C[\mathsf{exp}(g, x)]$ for a constant $x : \mathsf{xp}$; this represents a concrete half-key generated by an honest agent (with secret exponent x).*
- *$C[v]$ for a variable $v : \mathsf{hk}$; this represents that an agent receives some arbitrary value v as a Diffie-Hellman half-key (and our theorems below will imply that we can make the type restrictions here).*
- *$\mathsf{exp}(g, x)$ for constant $x : \mathsf{xp}$ without the surrounding context $C[\cdot]$ may only occur on the top, i.e., as t or as an element of M in the constraint $M \rhd t$.*
- *v for a constant $v : \mathsf{hk}$ without the surrounding context $C[\cdot]$ may again only occur on the top.*

– $\mathsf{exp}(\mathsf{exp}(g,x),y)$ *for constants* $x,y:\mathsf{xp}$*; this may only occur in a key position or in the form* $M \rhd \mathsf{exp}(\mathsf{exp}(g,x),y)$ *(when the intruder attempts to generate the key). It represents a concrete Diffie-Hellman key generated by two honest agents.*
– $\mathsf{exp}(v,y)$ *for variable* $v:\mathsf{hk}$ *and constant* $y:\mathsf{xp}$*. Again this may only occur in a key position or as* $M \rhd \mathsf{exp}(v,y)$*.*
– *The generator* g *is part of the intruder knowledge in every conjunct.*

Moreover, the entire constraint may not contain any non-variable subterm other than of form $C[\mathsf{exp}(g,x)]$ *or* $C[v]$ *for* $v:\mathsf{hk}$ *that can be unified with* $C[z]$ *for a fresh variable* z*.*

Finally, we also assume that an agent cannot be "told" a symmetric key, i.e. we never have $\mathsf{scrypt}(k,m)$ *for a variable* k*.*

Example 4. The constraint of the previous example is well-formed for Diffie-Hellman. Here, the context is $C[\cdot]=\mathsf{scrypt}(k,\cdot)$, and note that there is no other non-variable subterm of the constraint that can be unified with $C[z]$ for a fresh variable z. Suppose, however, the protocol would also contain another message encrypted with k, say $\mathsf{scrypt}(k,[A,B]_2)$, then there is a unifier with $C[\cdot]$, so $[A,B]_2$ could potentially be mistaken by an honest agent as a Diffie-Hellman half-key. In most cases such a type-confusion cannot be exploited by the intruder. In general, however, such type-confusions can be the cause of attacks that do not have a well-typed counterpart (e.g. think of $\mathsf{scrypt}(k,g)$). For our result it is thus necessary to exclude at least the type-confusions about half-keys. In general, a good strategy is some unique identifier (tag) into messages to avoid confusion, for instance $C[\cdot]=\mathsf{scrypt}(k,[hk,\cdot]_2)$ for a constant hk. Which of the many ways to distinguish half-keys from other messages is however not prescribed by our approach.

The well-formedness constraints are an important step towards our result, requiring that all exponents can only occur in a way this would done for Diffie-Hellman, in particular the only form in which the secret exponents of honest agents "get" into a message is for the half-keys of form $\mathsf{exp}(g,x)$ and for full-keys $\mathsf{exp}(v,x)$ as an encryption key. We can immediately derive an important semantic property from this, namely that the intruder cannot ever derive any secret exponent of an honest agent, or an exponentiation where more than one exponent is from an honest agent:

Lemma 2. *Consider a constraint* ϕ *that is well-formed for Diffie-Hellman and where* X *is the set of all constants of type* xp *(i.e. all constants that honest agents have generated). Consider any model* $\mathcal{I}$ *of* ϕ*, and any set* M *that occurs as intruder knowledge in* ϕ*. Then* $\mathcal{I}(M) \not\rhd x$ *for any* $x \in X$*; moreover, for any term* b*,*

$$\mathcal{I}(M) \not\rhd \mathsf{exp}(\ldots(\mathsf{exp}(b,x_1),\ldots),x_n)$$

whenever more than one $x_i \in X$*.*

Proof. The first part of the statement follows from the fact that we do not have any analysis rule for exp: from the (ground) initial intruder knowledge, thus no element of X can be derived, and it follows for every larger extension of the intruder knowledge, because the interpretation of variables cannot contain any element of X other than as the second argument of an exp.

The second part stems from the fact that the only ways an honest agent transmits messages containing a secret exponent x is

- $\mathsf{exp}(g, x)$ which contains only one secret exponent and
- $\mathsf{scrypt}(\mathsf{exp}(v, x), m)$ from which the $\mathsf{exp}(v, x)$ cannot be obtained, because a key cannot be learned from an encryption that uses that key.

Note that for the $\mathsf{exp}(g, x)$ case, the intruder may apply exp to such the term with self-generated exponents as much as he likes, and by the algebraic property, these exponents can be commuted.

Again this is immediate for the ground initial intruder knowledge, and carries over to later constraints, because all occurring variables are interpreted by a term generated from a previous knowledge. □

The analysis rules can destroy the well-formedness property: when applying analysis to the first conjunct of

$$\{[m_1, m_2]\} \rhd m_1 \wedge \{[m_1, m_2], m_3\} \rhd m_2$$

we obtain

$$\{[m_1, m_2], m_1, m_2\} \rhd m_1 \wedge \{[m_1, m_2], m_3\} \rhd m_2$$

which is not well-formed. This can be avoided by applying analysis steps for a message s to all constraints that contain s in the knowledge, starting with the largest knowledge. We call this the *analyze-all strategy* (and we later show that constraint reduction is complete under this strategy).

Lemma 3. *Under the analyze-all strategy, backwards applying rules preserves well-formedness for Diffie-Hellman.*

Proof. The standard well-formedness (as of Definition 2) is straightforward except for the analyze-all strategy: here we just need to ensure that analysis is applied to the last constraint that first (the one with the largest knowledge).

For the exponentiation/Diffie-Hellman related properties, we need to consider the rules individually.

Unify Consider the unification of two terms s and t where one contains one of the critical subterms:

- $C[\mathsf{exp}(g, x)]$. This only works if the other term has at the corresponding position a subterm u that can be unified with $C[\mathsf{exp}(g, x)]$. If u is a variable, so if the unifier maps u to $C[\mathsf{exp}(g, x)]$, still all "new" occurrences of $C[\mathsf{exp}(g, x)]$ satisfy the property. Otherwise, u is a non-variable subterm that unifies with $C[\cdot]$ and thus has to be itself of the form $C[z]$. Thus, z can only be $\mathsf{exp}(g, x)$ or a variable of type hk (so hk is matched with $\mathsf{exp}(g, x)$). In both cases the well-formedness is preserved.

– $C[v]$. This case is similar to the previous.
– $v : \mathsf{hk}$ without context: can only occur in a top position, which is excluded from unification by the *admitted*$(\cdot)$ predicate.
– $\mathsf{exp}(g, x)$, $x : \mathsf{xp}$ without context: can only occur in a top position. Again due to admittance, the only other term that can be unified has to be of the form $\mathsf{exp}(t_1, t_2)$ and t_1 can neither be a variable or an exponentiation. Thus $t_1 = g$ and $t_2 = x$ is the only case that can occur, which preserves well-formedness.
– $\mathsf{exp}(\mathsf{exp}(g, x), y)$. When appearing in a key-position, then the matching occurrence in the other term–because that cannot be a variable–also has to be an exponentiation, and then it can only be of the form $\mathsf{exp}(\mathsf{exp}(g, u), v)$ or $\mathsf{exp}(u, z)$ with appropriate types, so the result is still well-formed. When it appears on top level as $M \rhd \mathsf{exp}(\mathsf{exp}(g, u), v)$, then by well-formedness, there cannot be a term $s \in M$ that can be unified with our rule (s variable excluded, exponentiation $\mathsf{exp}(g, z)$ does not unify).
– $\mathsf{exp}(v, x)$. The case for a key-position is similar as in the previous item, and for $M \vdash \mathsf{exp}(v, x)$, unification is not admitted.

Generate cannot be applied to a variable or a term that has exp as root symbol; applying it to $C[m]$ yields m (which is well-formed both possible cases $m = \mathsf{exp}(g, x)$ and $m = v$).

Analyze can be used to decipher a term encrypted with a Diffie-Hellman key $\mathsf{exp}(\mathsf{exp}(g, x), y)$ or $\mathsf{exp}(v, x)$, which in both cases leaves a well-formed constraint (for the key derivation). □

4.4 Completeness

We now have everything in place to show that our reduction procedure will find an attack if there is one, given a constraint that is well-formed for Diffie-Hellman.

Theorem 1. *Given a satisfiable constraint ϕ that is well-formed for Diffie-Hellman, then a simple constraint is reachable from ϕ using the constraint reduction rules, proving that ϕ is satisfiable.*

Proof. Since the given constraint ϕ is satisfiable, exists a model $\mathcal{I}$ such that for every conjunct $M \rhd m$ of ϕ it holds that $\mathcal{I}(M) \rhd \mathcal{I}(m)$. We can easily regard the derivation of $\mathcal{I}(m)$ as a "proof tree" with $\mathcal{I}(m)$ as a root, the inductive rules of $\rhd$ (Definition 1) as construction steps, and messages from $\mathcal{I}(M)$ as leaves. In the following we thus assume that every term m to derive in ϕ is appropriately labeled with such a tree. The proof argument is now that either ϕ is simple or there is at least one constraint reduction step we can do that is compatible with interpretation $\mathcal{I}$. Since we have not made any particular assumptions about $\mathcal{I}$, every satisfiable constraint must therefore allow for a reduction to a simple constraint. The reduction of ϕ to a simple constraints proves the satisfiability of ϕ since the rules are sound (Lemma 1) and a simple constraint is satisfiable.

So if ϕ is simple, we are already done. So assume ϕ is not simple. Let $M \rhd t$ be the "first non-simple" conjunct of ϕ, i.e., one such t is not a variable and if $t = \mathsf{exp}(v, x)$ for a variable v, then $\mathsf{exp}(g, x) \notin M$. Considering the first operation of the derivation tree for $\mathcal{I}(t)$, we choose what to do next:

– Generate step and top symbol of t is not an exponentiation: use the generate rule of the lazy intruder and label the constraints for the subterms of t with the appropriate subtrees of t's derivation tree.
– Generate step and $t = \mathsf{exp}(g, x)$ or cannot occur: this would mean the intruder can generate x, then ϕ cannot be satisfiable by Lemma 2.
– Generate step and $t = \mathsf{exp}(\mathsf{exp}(g, x), y)$ similar to previous step.
– Generate step and $t = \mathsf{exp}(v, x)$ for a variable $v : \mathsf{xp}$. The syntactical decomposition (i.e. the intruder knowing both v and x and applying exp) is again absurd as in the previous cases. Thus $\mathcal{I}(v) \approx \mathsf{exp}(t_1, t_2)$ and the intruder can generate both $\mathsf{exp}(t_1, x)$ and t_2. (Note that t_1 should be g in a "well-typed" attack, but the intruder may have applied his exponentiations to an initially given $\mathsf{exp}(g, x)$ so that we get a more complex term that has x in the exponents.) By Lemma 2 we can conclude that t_1 cannot contain any secret exponents. Moreover $\mathsf{exp}(t_1, x)$ is derivable from $\mathcal{I}(M)$. We can also conclude that this derivation of $\mathsf{exp}(t_1, x)$ contains as a node the term $\mathsf{exp}(g, x)$, because (again by Lemma 2) this is the only way the intruder can get hold of a term that contains a secret exponent. We now look at the derivation of this $\mathsf{exp}(g, x)$.

  If the derivation of this $\mathsf{exp}(g, x)$ is a leaf node, then there is a term $t_0 \in M$ such that $\mathcal{I}(t_0) = \mathsf{exp}(g, x)$. If $t_0 = \mathsf{exp}(g, x)$, this conjunct is already simple (in contrast to our assumption). The cases $t_0 = \mathsf{exp}(v, x)$ and $t_0 = \mathsf{exp}(g, v)$ for a variable v cannot occur thanks to well-formedness. The remaining case that t_0 itself is a variable means, by well-formedness, that there is an earlier constraint (with smaller intruder-knowledge) where t_0 could be derived, and we can just proceed with the labeling of that occurrence of t (which puts us into one of the other cases).

  If the derivation of this $\mathsf{exp}(g, x)$ is an analysis node, then we perform use the Ana rule of the constraint reduction for this constraints and all other constraints that have at least knowledge M. We discuss the case of analysis steps below in detail and this case works exactly as those. The result of the analysis step preserves the $\mathcal{I}$ and gets us closer to a simple constraint.
– This case care of the generate case, since we have handled all possible occurrences of exponentiations for generate.
– The derivation tree is a leaf node (so $\mathcal{I}(t) \in \mathcal{I}(M)$) and top symbol is not exp. There must be an $s \in M$ such that $\mathcal{I}(s) = \mathcal{I}(t)$. If s is not a variable, we can thus apply the Unify rule and have successfully showed that there is a reduction step we can make that preserves $\mathcal{I}$. If s is a variable, then by well-formedness and the fact that we picked the first non-simple constraint, there is an earlier constraint $M_0 \rhd s$ (for some $M_0 \subseteq M$) and we can proceed with the derivation tree that labels the derivation of s here (and obtain one of the other cases).
– The derivation tree is a leaf node and $t = \mathsf{exp}(g, x)$. Again, there is a term $s \in M$ such that $\mathcal{I}(s) = t$. If $s = \mathsf{exp}(g, x)$ the Unify rule can be applied (and we are done). The cases $s = \mathsf{exp}(g, v)$ and $s = \mathsf{exp}(v, x)$ for a variable v are excluded by well-formedness. In the case that s is a variable itself, again

by well-formedness and picking the first non-simple constraint means that there is an earlier constraint upon which we can follow.

– The derivation tree is a leaf node and $t = \mathsf{exp}(v, x)$. This case is similar to the previous one, only if the unifiable term $s = \mathsf{exp}(g, x) \in M$ is ground, then this constraint is simple (because $\mathsf{exp}(g, x) \in M$), in contrast to our assumption, the other cases are already absurd.
– The derivation tree is a leaf node and $t = \mathsf{exp}(\mathsf{exp}(g, x), y)$ would mean the constraint is not satisfiable (by Lemma 2).
– The derivation tree has an analysis node at the top. We follow the analyzed term in the derivation tree:

  - if that is itself an analysis step, we recursively proceed with that analysis step, until we find either a generate or leaf.
  - if we reach a generate step, then we have a redundancy in the proof tree: the intruder first composed a term and then decomposed it again. We can simplify the proof tree in this case obviously, and arrive at some other case.
  - if we reach a leaf, we can analyze the respective term using the Ana reduction rule.

  So we have to now perform an analysis step of a message in $\mathcal{I}(M)$, let $s \in M$ be the respective symbolic message. If s is a variable, then by well-formedness we know s occurs in an earlier constraint that must be simple (because we picked the first non-simple constraint). In this case, we can simply replace the leaf node for $\mathcal{I}(s)$ with the respective derivation tree of s in the earlier constraint and end up in any of the other cases.

  If s is thus not a variable, then the Ana reduction rule can be applied, and the constraints for the key terms are labeled with the respective sibling trees of s in the derivation tree.

We have thus shown that every satisfiable well-formed non-simple constraint admits the application of a reduction rule that preserves its satisfiability. (Technically, the proof often referred to other cases, when there are redundancies like decomposition of a self-composed term, but that is only an intermediate step in the search for a reduction step.)

It remains to show that our reductions will eventually reach a simple constraint. We do not need to show the general termination of the procedure (as it would be necessary when using the lazy intruder as an analysis technique), but it suffices to show only that the reductions that our proof will use do not go into an infinite loop. This follows immediately from the construction that we assume the constraint to be labeled with the Dolev-Yao deduction trees for one ground solution. Every node of this tree will lead to at most one reduction in our argument, and thus this is finite. This concludes that every well-formed, satisfiable constraint admits reduction to a simple satisfiable form, and thus the reduction is complete. □

5 Millen's Minimal Diffie-Hellman Theory

We now apply the result of Theorem 1 and illustrate how drastically we can limit the intruder without losing completeness. Millen [8, 17] introduced a simple theory for modeling Diffie-Hellman using two new function symbols *kap* and *kas* to abstract the two Diffie-Hellman related operations, namely constructing the half key $kap(x)$ from secret x and constructing the full key $kas(t, y)$ from half-key t and secret y, along with the algebraic property:

$$kas(kap(X), Y) \approx kas(kap(Y), X) \ .$$

While this is just an approximation that is not complete in general, we can show it is complete for protocols that produce only well-formed constraints. To that end, we define a translation from our well-formed constraints using exp to analogous constraints using *kap* and *kas* and give an adaption of the notions of admissibility, simplicity, and well-formedness for the new variant.

Definition 6. *Define the translation α that maps a constraint that is well-formed for Diffie-Hellman to a corresponding one using kap and kas by the following replacements:*

- *Replace every occurrence of $\mathsf{exp}(g, x)$ (for a constant $x : \mathsf{xp}$) with $kap(x)$.*
- *Every occurrence of a variable $v : \mathsf{hk}$ with a $kap(v')$ where $v' : \mathsf{xp}$ is a new variable (the same v' for every occurrence of the same v).*
- *Every occurrence of $\mathsf{exp}(kap(x), y)$ with $kas(kap(x), y)$.*

We can define an adapted notion of well-formedness for Diffie-Hellman in the kap-kas-representation simply as the α-image of well-formed constraints for Diffie-Hellman in the exp representation. This is equivalent to the following explicit definition on the kap-kas-representation:

Definition 7. *We say a constraint ϕ is* well-formed for Diffie-Hellman in kap-kas-representation *w.r.t. a half-key context $C[\cdot]$, if it is both well-formed in the sense of Definition 2, it does not contain exp, and every occurrence of kas and kap, of terms of type xp, hk, and of the context $C[\cdot]$ is in one of the following forms:*

- *$C[kap(x)]$ for a constant $x : \mathsf{xp}$; this represents a concrete half-key generated by an honest agent (with secret exponent x).*
- *$C[kap(v')]$ for a variable $v' : \mathsf{xp}$; this represents that an agent receives $kap(v')$ as a Diffie-Hellman half-key.*
- *$kap(x)$ for constant $x : \mathsf{xp}$ without the surrounding context $C[\cdot]$ may only occur on the top, i.e., as t or as an element of M in the constraint $M \rhd t$.*
- *$kap(v')$ for a constant $v' : \mathsf{xp}$ without the surrounding context $C[\cdot]$ may again only occur on the top.*
- *$kas(kap(x), y)$ for constants $x, y : \mathsf{xp}$; this may only occur in a key position or in the form $M \rhd kas(kap(x), y)$ (when the intruder attempts to generate the key). It represents a concrete Diffie-Hellman key generated by two honest agents.*

– $kas(kap(v'), y)$ *for variable* $v' : \mathsf{xp}$ *and constant* $y : \mathsf{xp}$*. Again this may only occur in a key position or as* $M \rhd kas(kap(v'), y)$*.*

Moreover, the entire constraint may not contain any non-variable subterm other than of form $C[kap(x)]$ *or* $C[kap(v')]$ *for* $x, v :' \mathsf{xp}$ *that can be unified with* $C[z]$ *for a fresh variable* z*.*

Definition 8. *Similarly we can carry over the notion of simplicity using* α*; explicitly that is the least relation* $simpleM(\cdot)$ *such that:*

– $simpleM(M \rhd v)$ *if* $v \in \mathcal{V}$
– $simpleM(M \rhd kap(v'))$ *if* $v' \in \mathcal{V}$ *(and* $v' : \mathsf{xp}$*)*
– $simpleM(\phi_1)$ *and* $simpleM(\phi_2)$ *implies* $simpleM(\phi_1 \wedge \phi_2)$*.*
– $simpleM(\phi)$ *implies* $simpleM(\phi \wedge (M_0 \rhd kap(v')) \wedge (M \rhd kas(kap(v'), c)))$ *provided that* $v' \in \mathcal{V}$*,* c *is a constant, and* $kap(c) \in M$*.*

Similarly adapting the admissibility predicate, we get $admittedM(t)$ *if* $t \notin \mathcal{V}$ *and* $t \neq kas(\cdot, \cdot)$*.*

Theorem 2. *Consider a satisfiable constraint* ϕ *that is well-formed for Diffie-Hellman. Then also* $\alpha(\phi)$ *is satisfiable (considering kap and kas as public symbols of* Σ_p*). Also, there exists a solution where all variables* $v' : \mathsf{xp}$ *are substituted for constants of type* xp*.*

With the adapted notions of well-formedness, simplicity and admissibility for kap and kas, also the constraint reduction procedure is still complete.

Proof. Observe that α is a homomorphism on the constraint reduction: Given a constraint ϕ that is well-formed for Diffie-Hellman; if we can derive ϕ' from ϕ with one reduction, then we can derive $\alpha(\phi')$ from $\alpha(\phi)$ with one reduction.

This in particular means that if from ϕ we can derive a simple constraint ϕ' then we can derive the simple $\alpha(\phi')$ from $\alpha(\phi)$. This gives us that $\alpha(\phi)$ is satisfiable, and that every $v' : \mathsf{xp}$ can be instantiated arbitrarily with an intruder-known term, in particular type-correctly with an (intruder-generated) $c : \mathsf{xp}$. (Note that of course the intruder can generate constants of type xp, but they never appear in the constraint reduction due to laziness.) Moreover, the adapted constraint reduction is thus correct on the Millen-variant of constraints that are well-formed for Diffie-Hellman. □

Note that this replacement is more restrictive than originally used in CAPSL and in the notion of well-formedness of the previous section: honest agents will only accept messages of the form $kap(x)$ or $kap(v')$ as half-keys—i.e. as if they can check this is indeed the result of an exponentiation. This is in fact a strong typing result.

5.1 Freedom of Diffie-Hellman

Using this restricted model of *kas* and *kap* and the "atomic arguments" of exponentiation, there is just one form of algebraic reasoning left, namely that

$\mathit{kas}(\mathit{kap}(t_1), t_2) \approx \mathit{kas}(\mathit{kap}(t_2), t_1)$ where t_1 and t_2 are either constants of type xp or variables which can only be instantiated with constants of type xp, but not with composed terms.

A first idea to avoid algebraic reasoning entirely, based on this restrictive model, is to rewrite a protocol description as follows. For every message that an honest agent can receive and that contains a Diffie-Hellman key $\mathit{kas}(\mathit{kap}(t_1), t_2)$, we add the variant of this message where the t_i are swapped, i.e. $\mathit{kas}(\mathit{kap}(t_2), t_1)$, so that the agent can accept either form. [3] This basically allows for performing unification of messages in the free algebra, instead of unification modulo the $\mathit{kas}/\mathit{kap}$-property. There a few examples that require some more care, namely when third parties are involved that do not know the key and therefore are "blind" for the format $\mathit{kas}(\mathit{kap}(\cdot), \cdot)$. We only stipulate here that these protocols can either be handled using more typing results or by making restrictions on the form of protocols considered.

This idea has the disadvantage that it blows up the descriptions, essentially making all potential alternatives of the algebraic reasoning explicit in the protocol description. We now show that we can do better for a large class of protocols. The idea is a *normalized* representation of Diffie-Hellman keys in the sense that we choose one of the two representations once and for all. More precisely, in the protocol description we will always order keys as $\mathit{kas}(\mathit{kap}(X), Y)$ where X is a variable that represents the secret of the *initiator* role and Y represents the secret of the *responder* role. Thus, even though the initiator would actually construct the Diffie-Hellman key as $\mathit{kas}(\mathit{kap}(Y), X)$, the protocol description uses the $\mathit{kas}(\mathit{kap}(X), Y)$ representation on his side. However, interpreting such a protocol representation in the free algebra can exclude attacks in general, because the concrete Diffie-Hellman half-keys alone do not tell whether they belong to the initiator or the responder role. Consider the exchange protocol:

$$\begin{aligned} A \to B &: \mathsf{m}_1, \mathsf{sign}(\mathsf{inv}(\mathsf{pk}(A)), [B, \mathit{kap}(X)]_2) \\ B \to A &: \mathsf{m}_2, \mathsf{sign}(\mathsf{inv}(\mathsf{pk}(B)), [A, \mathit{kap}(Y)]_2) \end{aligned}$$

where m_i are (commonly known) tags, and consider the trace:

$$\begin{aligned} a \to b(i) &: \mathsf{m}_1, \mathsf{sign}(\mathsf{inv}(\mathsf{pk}(a)), [b, \mathit{kap}(x_1)]_2) \\ b \to a(i) &: \mathsf{m}_1, \mathsf{sign}(\mathsf{inv}(\mathsf{pk}(b)), [a, \mathit{kap}(x_2)]_2) \\ b(i) \to a &: \mathsf{m}_2, \mathsf{sign}(\mathsf{inv}(\mathsf{pk}(b)), [a, \mathit{kap}(x_2)]_2) \\ a(i) \to b &: \mathsf{m}_2, \mathsf{sign}(\mathsf{inv}(\mathsf{pk}(a)), [b, \mathit{kap}(x_1)]_2) \end{aligned}$$

After this trace, a and b have a shared secret key, but both of them believe to be playing as the initiator role A. Thus, when we normalize key terms by the initiator role, a would use the key $\mathit{kas}(\mathit{kap}(x_1), x_2)$ (because it believes b to be responder) and vice-versa b would use the key $\mathit{kas}(\mathit{kap}(x_2), x_1)$. So by the confusion about the roles, the keys would still syntactically differ and free-algebra reasoning would not be sound in this case.

[3] If there are several occurrences of Diffie-Hellman keys in a message, there is an exponential number of variants to consider.

5.2 Authenticating the Role

Suppose the tag $\mathsf{m_i}$ in the above example protocol were part of the signature, i.e. authenticating not only the half-key and intended recipient itself, but also the role, then the role confusion can not arise and we can use the free algebra interpretation of the normalized form.

To formalize this idea, we consider again the concept of a context $C[\cdot]$ that surrounds every exchange of half-keys of § 4.1. From now on, we use two different contexts that distinguish initiator and responder role, e.g. in the example:

$$\begin{aligned} C_1(B)[\cdot] &= [\mathsf{m}_1, B, \cdot]_3 \\ C_2(A)[\cdot] &= [\mathsf{m}_2, A, \cdot]_3 \end{aligned}$$

Note that here we use as context only the immediate surrounding constructor, i.e. the concatenation, and not yet the authentication. The reason is that our previous theorems rely on a one-level context and would not hold for many-level contexts. The modification from one to two contexts, is not a problem, however, adapting all definitions to have either C_1 or C_2 in places where we previously had only C. The adapted well-formedness for Diffie-Hellman thus in particular requires that (for any term t) neither $C_1[t]$ nor $C_2[t]$ can be unified with any other non-atomic subterm of the protocol messages. Additionally, we require that for any term s and t, $C_1[s]$ has no unifier with $C_2[t]$, i.e. the two roles are always uniquely distinguished.

For the authentication, we consider a further pair of contexts $C_1^A[\cdot]$ and $C_2^A[\cdot]$ (which we allow to be unifiable with each other and with other message parts); in our example we have $C_1^A(A)[\cdot] = C_2^A(A)[\cdot] = \mathsf{sign}(\mathsf{inv}(\mathsf{pk}(A)), \cdot)$.

In addition to the previous requirements for well-formedness, we now require:

1. For every full-key of the form $kas(kap(t_1), t_2)$, the half-keys occur in the constraints as $C_1^A[C_1[kap(t_1)]]$ and $C_2^A[C_2[kap(t_2)]]$ and only in this way.
2. Given any occurrence of $C_i^A[C_i[kas(t)]]$ where C_i^A can be constructed by the intruder using the generate rule, then t is a variable.
3. An encryption with a Diffie-Hellman key cannot occur as a subterm of another encryption.

Requirement 2 ensures the authentication of the roles: he cannot generate the term that corresponds to the authenticated half-key (and thereby determine the role) of an honest agent (where t would be a constant). In our example, we could define the initial intruder knowledge to contain $\mathsf{inv}(\mathsf{pk}(i))$ but no other private keys, so the intruder can generate the signatures for the Diffie-Hellman half-keys (and determine the role) iff he is acting under his real name i.

Theorem 3. *Given a constraint (for Millen's theory) that satisfies all said requirements. Then this constraint is satisfiable in Millen's theory iff it is satisfiable in the free algebra.*

Proof. We show that the reduction of a constraint ϕ to a simple constraint can always be achieved without the use of algebraic properties: we never need to perform a “cross-unification” of two Diffie-Hellman keys $kas(kap(t_1), t_2)$ and $kas(kap(t_3), t_4)$—i.e. one where we unify t_1 with t_4 and unify t_2 with t_3. Such a “cross-unification” would of course not be found when interpreting terms in the free algebra.

So consider a Unify-step in the reduction of ϕ that involves the unification of two such Diffie-Hellman keys. By the well-formedness conditions for the kap-kas-representation, all the t_i are constants or variables of type xp and, by condition 1, the original constraint ϕ must contain the four half-keys and always with their correct contexts, namely $C_1^A[C_1[kap(t_1)]], C_2^A[C_2[kap(t_2)]], C_1^A[C_1[kap(t_3)]]$, and $C_2^A[C_2[kap(t_4)]]$; the first two belong to the full key $kas(kap(t_1), t_2)$ and the second two belong to the full-key $kas(kap(t_3), t_4)$.

One of the first two represents the half-key generated by an honest agent, so either t_1 or t_2 must be a constant, and the other is part of a term that the intruder has to generate (and that t_i may either be a constant or a variable). The same holds for the other two half-keys.

By condition 2, if the intruder can compose the context $C_i^A[\cdot]$ of a half key $kap(t_j)$ (using the Generate rule in the constraint reduction) then the t_j must be a variable (for the original constraint ϕ; it may be instantiated during reduction). Thus, if any t_j that the intruder has to generate are constants, then the respective $C_i^A[C_i[kap(t_j)]]$ can only have been obtained using a Unify step. According to the well-formedness, they can only belong to the role as which they were meant by the agent who created them—excluding the “cross-unification” (t_1 with t_4 and t_2 with t_3). So the only remaining possibility for a cross-unification is that both intruder-determined half-keys—say t_1 and t_3 (the proof is identical for the other 3 cases)—are variables and the intruder generates the context using the generate rule.

So the intruder needs to generate a term m that contains $\mathsf{scrypt}(kas(kap(v), x), m_0)$ and we have a reduction involving Unification with another message m' in the intruder knowledge that has as the respective subterm $\mathsf{scrypt}(kas(kap(v'), x'), m_0')$. (Again, x and x' are constants, v and v' are variables.) The half-key generation constraints are reduced to the simple constraints $M_1 \rhd kap(v)$ and $M_2 \rhd kap(v')$ during derivation. Now the messages m and m' can only use tuples as a context of the $\mathsf{scrypt}(\cdot, \cdot)$ message by condition 3. Thus, there exists an alternative derivation instead of applying the unify rule to m and m': the intruder can first analyze m_0' (choosing a known value as v) and then apply the generate rule to obtain m from m_0 (choosing a known value as v'). Intuitively, this case says that when the intruder runs two sessions with honest agents (the creators of x and x') then it is no restriction to assume that the intruder always uses self-generated half-keys for his part, rather than re-using half-keys (to which he does not know the secret) from other sessions. This however needs the condition that Diffie-Hellman encryptions never occur under another encryption to which the intruder has no access. $\square$

6 Conclusions and Related Work

Several works consider the integration of algebraic reasoning into protocol verification, in particular [6] extending the lazy intruder technique [19, 21, 4]. While [6] presents a verification method for exponentiation-based protocols in general (for a bounded number of sessions), our paper establishes a completely different kind of result, namely one that allows to avoid most or all of the algebraic reasoning. This result is not linked to a particular verification method, and works for an unbounded number of sessions. In fact, we use the lazy intruder as a convenient way to derive the results and formulating the class of protocols that we can support.

This paper is in the tradition of a number of relative soundness results which show that under certain conditions, models can be restricted without losing attackability, i.e. if there is an attack in the unrestricted model, then so is one in the restricted model. In particular, [12] justifies a strictly typed model (in the free algebra), [18, 15] show that one can safely avoid cancellation properties (and use free-algebra pattern matching), [14] show that one can simplify the algebraic theories used for Diffie-Hellman. Results like [16] show that the typing result of [12] can also be established under many algebraic theories and like us also use the lazy intruder technique for this. Our work compares to these soundness results by providing the most restrictive version for the Diffie-Hellman protocols so far, namely establishing the soundness of Millen's restrictive model for Diffie-Hellman [8, 17].

This very restrictive theory allows tools that support algebraic reasoning to avoid many unnecessary reasoning steps. We also sketch how to exploit this in tools that do not support algebraic reasoning at all. This is similar to the result of [13] which is however focused on ProVerif and its abstract model (that does not have fresh nonces).

Like the cited works for relative soundness, our result also relies on making restrictions on the class of protocols that are supported, in our case that Diffie-Hellman half-keys can be distinguished from other protocol parts. In fact, it is good engineering practice [1] that messages somehow identify what they mean, e.g. by tags. One can thus rephrase this and other works as *exploiting properties of well-designed protocols to simply their verification.*

Currently, our result supports only protocols that use Diffie-Hellman in its classical form and does not cover protocols where for instance the exponents contain both ephemeral and long-term secrets. We also did not consider the algebraic properties for the inversion of exponents (because this is irrelevant for the classical Diffie-Hellman). We believe that the extension of our results to such broader classes of protocols and algebraic theories is possible and plan to investigate this in the future.

References

1. M. Abadi and R. M. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Trans. Software Eng.*, 22(1):6–15, 1996.

2. M. Arapinis and M. Duflot. Bounding messages for free in security protocols. In V. Arvind and S. Prasad, editors, *FSTTCS'07*, volume 4855 of *LNCS*, pages 376–387, New Delhi, India, Dec. 2007. Springer.
3. A. Armando and L. Compagna. SAT-based Model-Checking for Security Protocols Analysis. *Int. J. of Information Security*, 6(1):3–32, 2007.
4. D. A. Basin, S. Mödersheim, and L. Viganò. OFMC: A symbolic model checker for security protocols. *Int. J. Inf. Sec.*, 4(3):181–208, 2005.
5. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96, Cape Breton, Nova Scotia, Canada, June 2001. IEEE Computer Society.
6. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents. In P. K. Pandya and J. Radhakrishnan, editors, *FSTTCS*, volume 2914 of *Lecture Notes in Computer Science*, pages 124–135. Springer, 2003.
7. C. J. F. Cremers. The Scyther tool: Verification, falsification, and analysis of security protocols. In *CAV*, pages 414–418, 2008.
8. G. Denker and J. Millen. CAPSL and CIL Language Design. Technical Report SRI-CSL-99-02, SRI, 1999.
9. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
10. S. Escobar, C. Meadows, and J. Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In A. Aldini, G. Barthe, and R. Gorrieri, editors, *FOSAD*, volume 5705 of *Lecture Notes in Computer Science*, pages 1–50. Springer, 2007.
11. D. Harkins and D. Carrel. The Internet Key Exchange (IKE), 1998. IETF, RFC 2409.
12. J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. *Journal of Computer Security*, 11(2):217–244, 2003.
13. R. Küsters and T. Truderung. Using ProVerif to analyze protocols with Diffie-Hellman exponentiation. In *CSF*, pages 157–171, 2009.
14. C. Lynch and C. Meadows. Sound approximations to Diffie-Hellman using rewrite rules. In J. Lopez, S. Qing, and E. Okamoto, editors, *ICICS*, volume 3269 of *Lecture Notes in Computer Science*, pages 262–277. Springer, 2004.
15. C. Lynch and C. Meadows. On the relative soundness of the free algebra model for public key encryption. *Electr. Notes Theor. Comput. Sci.*, 125(1):43–54, 2005.
16. S. Malladi. Protocol indepedence through disjoint encryption under exclusive-or. In *TOSCA 2011*, volume 6993 of *LNCS*, 2011.
17. J. Millen and F. Muller. Cryptographic Protocol Generation From CAPSL. Technical Report SRI-CSL-01-07, SRI, 2001.
18. J. K. Millen. On the freedom of decryption. *Inf. Process. Lett.*, 86(6):329–333, 2003.
19. 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.
20. S. Mödersheim, L. Viganò, and D. A. Basin. Constraint differentiation: Search-space reduction for the constraint-based analysis of security protocols. *Journal of Computer Security*, 18(4):575–618, 2010.
21. M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions, composed keys is NP-complete. *Theor. Comput. Sci.*, 1-3(299):451–475, 2003.
22. M. Turuani. The CL-Atse protocol analyser. In *RTA*, pages 277–286, 2006.