Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents*

Yannick Chevalier[1], Ralf Küsters[2], Michaël Rusinowitch[1], Mathieu Turuani[2]

[1] LORIA-INRIA-Universités Henri Poincaré, France
email:{chevalie,rusi,turuani}@loria.fr

[2] Department of Computer Science, Stanford University, USA
email: kuesters@theory.stanford.edu

Abstract. We present an NP decision procedure for the formal analysis of protocols in presence of modular exponentiation with products allowed in exponents. The number of factors that may appear in the products is unlimited. We illustrate that our model is powerful enough to uncover known attacks on the A-GDH.2 protocol suite.

1 Introduction

Most automatic analysis techniques for security protocols take as a simplifying hypothesis that the cryptographic algorithms are perfect: One needs the decryption key to extract the plaintext from the ciphertext, and also, a ciphertext can be generated only with the appropriate key and message (no collision). Under these assumptions and given a bound on the number of protocol sessions, the insecurity problem is decidable (see, e.g., [1]). However, it is an open question whether this result remains valid when the intruder model is extended to take into account even simple properties of product or exponentiation operators, such as those of RSA and Diffie-Hellman Exponentiation. This question is important since many security flaws are the consequence of these properties and many protocols are based on these operators (see, e.g., [9]).

Only recently the perfect encryption assumption for protocol analysis has been slightly relaxed. In [7], unification algorithms are designed for handling properties of Diffie-Hellman cryptographic systems. Although these results are useful, they do not solve the more general insecurity problem. In [5, 6], decidability of security has been proved for protocols that employ exclusive or. When the XOR operator is replaced by an abelian group operator, decidability is mentioned as an open problem by [8]. In [8], there is a reduction from the insecurity problem to solving quadratic equation systems. However, the satisfiability of these systems is in general undecidable. Hence, this approach fails to solve the initial insecurity problem.

* This work was partially supported by PROCOPE and IST AVISPA. The second author was also supported by the DFG.

In this paper, using non-trivial extensions of our technique in [5], we show that the insecurity problem for protocols that use Diffie-Hellman exponentiation with products in exponents is NP-complete. Our model is powerful enough to uncover attacks on the A-GDH.2 protocol suite discovered in [9].

Boreale and Buscemi [3] have addressed a similar problem. However, in their paper among other restrictions they put an a priori bound on the number of factors that may occur in products. In the present paper we allow an unlimited number of factors. Also, Boreale and Buscemi do not provide a complexity result. Diffie-Hellman exponentiation is also studied by Millen and Shmatikov [8]. Similar to our work, Millen and Shmatikov assume that products only occur in exponents. However, they do not provide a decision procedure. Also, they assume the base in exponentiations to be a fixed constant. This is a severe restriction since in general this rules out realistic attacks, even in case the protocol under consideration assumes a fixed basis.

Structure of the paper. In Section 2, we introduce our protocol and intruder model. The decidability result is presented in Section 3, including the description of the NP decision algorithm, and an overview of the proof. Also, we point out the main differences to our proof presented in [5] for XOR. A very brief sketch of the main new part of the proof compared to the case for XOR is provided in Section 4. To illustrate our model, in Section 5 we formally specify the A-GDH.2 protocol and present an attack on it discovered by Pereira and Quisquater.

Full proofs of all results presented here can be found in our technical report [4].

2 The Protocol and Intruder Model

The protocol and intruder model we describe here extends standard models for automatic analysis of security protocols in two respects. First, messages can be build using the operator $Exp(\cdot,\cdot)$, which stands for Diffie-Hellman exponentiation, and a product operator "$\cdot$" for multiplication in an abelian group. Second, in addition to the standard Dolev-Yao intruder capabilities, the intruder is equipped with the ability to perform Diffie-Hellman exponentiation. In what follows, we provide a formal definition of our model by defining terms, messages, protocols, the intruder, and attacks.

2.1 Terms and Messages

The set of terms *term* is defined by the following grammar:

$$\begin{aligned} \mathit{term} &::= \mathcal{A} \mid \mathcal{V} \mid \langle \mathit{term}, \mathit{term} \rangle \mid \{\mathit{term}\}^s_{\mathit{term}} \mid \{\mathit{term}\}^p_{\mathcal{K}} \mid \mathit{Exp}(\mathit{term}, \mathit{product}) \\ \mathit{product} &::= \mathit{term}^{\boldsymbol{Z}} \mid \mathit{term}^{\boldsymbol{Z}} \cdot \mathit{product} \end{aligned}$$

where $\mathcal{A}$ is a finite set of constants (*atomic messages*), containing principal names, nonces, keys, and the constants 1 and secret; $\mathcal{K}$ is a subset of $\mathcal{A}$ denoting the set of public and private keys; $\mathcal{V}$ is a finite set of variables; and $\boldsymbol{Z}$ is the set of integers, the *product exponents*. We assume that there is a bijection $\cdot^{-1}$ on

$\mathcal{K}$ which maps every public (private) key k to its corresponding private (public) key k^{-1}. The binary symbols $\langle\cdot,\cdot\rangle$, $\{\cdot\}^s_\cdot$, and $\{\cdot\}^p_\cdot$ are called *pairing*, *symmetric encryption*, and *asymmetric encryption*, respectively. Note that a symmetric key can be any term and that for asymmetric encryption only atomic keys (namely, public and private keys from $\mathcal{K}$) are allowed. The product operator "$\cdot$" models multiplication in an abelian group, e.g., a subgroup G of order q of the multiplicative group $\mathbb{Z}_p^*$ where p and q are prime numbers (as in the A-GDH.2 protocol). Therefore, terms and products are read modulo commutativity and associativity of the product operator as well as the identity $t^1 = t$, and are written without parentheses. For instance, if a, b, and c are atomic messages, then $a^2 \cdot b^3 \cdot c^{-2}$ and $a \cdot b \cdot a \cdot b \cdot c^{-1} \cdot b \cdot c^{-1}$ are considerd the same products where c^{-1} is the inverse of c. The operator $Exp(\cdot,\cdot)$ stands for Diffie-Hellman exponentiation. For instance, $Exp(a, b^2 \cdot c^{-1})$ is a raised to the power of $b^2 \cdot c^{-1}$. Note that products only occur in exponents.

If $t, t_1, \ldots, t_n$ are terms with $n \geq 2$, then we call a product of the form t^z for some $z \neq 1$ or a product of the form $t_1^{z_1} \cdots t_n^{z_n}$ a *non-standard term* where the $z, z_1, \ldots, z_n$ are integers. By abuse of terminology, we refer to a term or a product as a "term". We say *standard term* to distinguish a term from a non-standard term. Note that in a product of the form $t_1^{z_1} \cdots t_n^{z_n}$, the terms t_i are standard terms.

Variables are denoted by $x, y, \ldots$, terms are denoted by s, t, u, v, finite sets of terms are written $E, F, \ldots$, and decorations thereof, respectively. For a term t and a set of terms E, we refer to the set of variables occurring in t and E by $\mathcal{V}(t)$ and $\mathcal{V}(E)$, respectively. For some set S, we denote the *cardinality* of S by $\mathsf{Card}(S)$.

A *ground term* (also called *message*) is a term without variables. We use the expressions *standard* and *non-standard messages* as in the case of terms. A *(ground) substitution* is a mapping from $\mathcal{V}$ into the set of *standard* (ground) terms. The application of a substitution σ to a term t (a set of terms E) is written $t\sigma$ ($E\sigma$), and is defined as expected.

The set of *subterms* of a term t, denoted by $\mathcal{S}(t)$, is defined as follows:

- If $t \in \mathcal{A}$ or $t \in \mathcal{V}$, then $\mathcal{S}(t) = \{t\}$.
- If $t = \langle u, v\rangle$, $\{u\}^s_v$, or $\{u\}^p_v$, then $\mathcal{S}(t) = \{t\} \cup \mathcal{S}(u) \cup \mathcal{S}(v)$.
- If $t = Exp(u, t_1^{z_1} \cdots t_p^{z_p})$, then $\mathcal{S}(t) = \{t\} \cup \mathcal{S}(u) \cup \bigcup_i \mathcal{S}(t_i)$.
- If $t = t_1^{z_1} \cdots t_p^{z_p}$, then $\mathcal{S}(t) = \{t\} \cup \bigcup_i \mathcal{S}(t_i)$.

We define $\mathcal{S}(E) = \bigcup\{\mathcal{S}(t) \mid t \in E\}$. Note that $Exp(a, b^2 \cdot c^1)$ and $b^2 \cdot c^1 \cdot d^1$ are not subterms of $Exp(a, b^2 \cdot c^1 \cdot d^1)$. We define the set $\mathcal{S}_{ext}(t)$ of *extended subterms* of t to be $\mathcal{S}_{ext}(t) = \mathcal{S}(t) \cup \{M \mid Exp(u, M) \in \mathcal{S}(t)\}$. For instance, $b^2 \cdot c^1 \cdot d^1$ is an extended subterm of $Exp(a, b^2 \cdot c^1 \cdot d^1)$, but $b^2 \cdot c^1$ is not.

We define the *DAG size* $|t|_{dag}$ of t to be $\mathsf{Card}(\mathcal{S}_{ext}(t))$. Note that the DAG size does not take into account the size needed to represent the product exponents occurring in a term. We define the *product exponent size* $|t|_{exp}$ of t to be $\Sigma_{t_1^{z_1} \ldots t_n^{z_n} \in \mathcal{S}_{ext}(t)} |z_1| + \ldots + |z_n|$, where $|z_i|$ is the number of bits needed to represent the integer z_i in binary. Finally, we define the *size* $||t||$ of t to be $|t|_{dag} + |t|_{exp}$.

For a set of terms E the size is defined in the same way (replace t by E in the above definitions). Let $\mathcal{S}_{ext}(\sigma) = \{t \mid t \in \mathcal{S}_{ext}(\sigma(x)), x \in \mathcal{V}\}$ for a substitution σ. We define $|\sigma|_{dag} := |\mathcal{S}_{ext}(\sigma)|_{dag}$ to be the *DAG size* of σ, $|\sigma|_{exp} := |\mathcal{S}_{ext}(\sigma)|_{exp}$ to be the *product exponent size* of σ, and $||\sigma|| := |\sigma|_{dag} + |\sigma|_{exp}$ to be the *size* of σ.

We now formulate the algebraic properties of terms. Besides commutativity and associativity of the product operator we consider the following properties:

$$\begin{aligned} t^1 &= t & t \cdot 1 &= t \\ t^0 &= 1 & 1^z &= 1 \\ t^z \cdot t^{z'} &= t^{z+z'} & & \\ Exp(t,1) &= t & Exp(Exp(t,t'),t'') &= Exp(t,t' \cdot t'') \end{aligned}$$

where z and z' are integers. A *normal form* $\ulcorner t \urcorner$ of a term t is obtained by exhaustively applying these identities from left to right. Note that $\ulcorner t \urcorner$ is uniquely determined up to commutativity and associativity of the product operator. Two terms t and t' are *equivalent* if $\ulcorner t \urcorner = \ulcorner t' \urcorner$. The notion of normal form extends in the obvious way to sets of terms and substitutions.

Let us illustrate the notion of a normal form by some examples: If $a, b, c, d \in \mathcal{A}$, then

- $\ulcorner (a^2 \cdot b^1) \cdot b^{-2} \urcorner = a^2 \cdot b^{-1}$,
- $\ulcorner Exp(Exp(a, b^1 \cdot c^1), c^{-1} \cdot d^{-2}) \urcorner = Exp(a, b \cdot d^{-2})$, and
- $\ulcorner Exp(Exp(a, b^3 \cdot c^{-6} \cdot b^{-3}), c^6) \urcorner = a$.

The following lemma summarizes basic properties of normal forms.

Lemma 1. *For every term t, t', and substitution σ the following holds true:*

1. $|\ulcorner t \urcorner|_{dag} \leq |t|_{dag}$,
2. $|\ulcorner t \urcorner|_{exp} \leq |t|_{exp}$,
3. $||\ulcorner t \urcorner|| \leq ||t||$, *and*
4. $\ulcorner t\sigma \urcorner = \ulcorner \ulcorner t \urcorner \sigma \urcorner = \ulcorner t \ulcorner \sigma \urcorner \urcorner = \ulcorner \ulcorner t \urcorner \ulcorner \sigma \urcorner \urcorner$.

2.2 Protocols

The following definition is explained below.

Definition 1. *A* protocol rule *is of the form $R \Rightarrow S$ where R and S are standard terms.*

A protocol *P is a tuple $(\{R_i \Rightarrow S_i,\ i \in \mathcal{I}\}, <_{\mathcal{I}}, E)$ where E is a finite normalized set of standard messages with $1 \in E$, the* initial intruder knowledge, *$\mathcal{I}$ is a finite (index) set, $<_{\mathcal{I}}$ is a partial ordering on $\mathcal{I}$, and $R_i \Rightarrow S_i$, for every $i \in \mathcal{I}$, is a protocol rule such that*

1. *the (standard) terms R_i and S_i are normalized,*
2. *for all $x \in \mathcal{V}(S_i)$, there exists $j \leq_{\mathcal{I}} i$ such that $x \in \mathcal{V}(R_j)$, and*

3. *for every subterm* $Exp(t_1, t_2^{z_2} \cdots t_n^{z_n})$ *of* R_i*, there exists* $r \in \{1, \ldots, n\}$ *such that* $\mathcal{V}(t_l) \subseteq \cup_{j <_{\mathcal{I}} i} \mathcal{V}(R_j)$ *for every* $l \in \{1, \ldots, n\} \setminus \{r\}$.

A bijective mapping $\pi : \mathcal{I}' \to \{1, .., k\}$ *is called* execution ordering *for* P *if* $\mathcal{I}' \subseteq \mathcal{I}$*,* k *is the cardinality of* $\mathcal{I}'$*, and for all* i, j *we have that if* $i <_{\mathcal{I}} j$ *and* $\pi(j)$ *is defined, then* $\pi(i)$ *is defined and* $\pi(i) < \pi(j)$*. We define the* size *of* π *to be* k.

Given a protocol P, in the following we will assume that $\mathcal{A}$ is the set of constants occurring in P. We define $\mathcal{S}(P) := \mathcal{S}(E \cup \bigcup_{i \in \mathcal{I}} (R_i \cup S_i))$ to be the *set of subterms of* P and $\mathcal{V} := \mathcal{V}(P)$ to be the set of variables occurring in P. The *set of extended subterms* $\mathcal{S}_{ext}(P)$, the *DAG size* $|P|_{dag}$, the *product exponent size* $|P|_{exp}$, and the *size* $||P||$ of P are defined as expected.

Intuitively, when executing a rule $R_i \Rightarrow S_i$ and on receiving a (normalized) message m in a protocol run, it is first checked whether m and R_i match, i.e., whether there exists a ground substitution σ such that $\lceil m \rceil = \lceil R_i \sigma \rceil$. If so, $\lceil S_i \sigma \rceil$ is returned as output. We always assume that the messages exchanged between principals (and the intruder) are normalized—therefore, m is assumed to be normalized and the output of the above rule is not $S_i\sigma$ but $\lceil S_i \sigma \rceil$. This is because principals and the intruder cannot distinguish between equivalent terms, and therefore, they may only work on normalized terms (representing the corresponding equivalence class of terms). Finally, we note that since the different protocol rules may share variables, the substitution of some of the variables in R_i and S_i may be known already.

Condition 1. in the above definition is not a real restriction since due to Lemma 1, the transformation performed by a protocol rule and its normalized variant coincide.

Condition 2. guarantees that when with S_i an output is produced, the substitution for all variables in S_i are determined beforehand. Otherwise, the output of a protocol rule would be arbitrary since if in S_i a variables occurs for the first time, it could be mapped to any message.

Condition 3. guarantees that every single protocol rule can be carried out deterministically. For instance, this singles out protocol rules of the form $Exp(g, x \cdot y) \Rightarrow Exp(g, x \cdot y \cdot b)$ in case neither x nor y are known beforehand (i.e., the substitution of neither x nor y is determined by a previous application of a rule). Otherwise, when say the principal receives $Exp(a, a \cdot b)$, there are different ways of matching this term with $Exp(a, x \cdot y)$. In our technical report [4], we explain why condition 3. does not rule out deterministic protocols. In a nutshell, such protocols can be turned into equivalent protocols (w.r.t. attacks possible) with rules that can be carried out deterministically. Let us illustrate the transformation by a simple example protocol with one principal A performing two steps A_1 and A_2:

$$\begin{aligned} A_1 &: \quad Exp(g, x \cdot y) \Rightarrow hello \\ A_2 &: \quad\quad\quad\quad y \Rightarrow Exp(g, x) \end{aligned}$$

This protocol does not satisfy condition 3. because neither x nor y are known when executing A_1. However, although A_1 cannot be performed deterministically, overall the protocol is deterministic since in A_2 principal A gets to know

y, and thus, can determine $Exp(g, x)$. This protocol can be turned into the following protocol:

$$\begin{array}{lll} A_1 & : & z \Rightarrow hello \\ A_2 & : & y \Rightarrow Exp(z, y^{-1}) \end{array}$$

Every rule in this description of A can be carried out deterministically without having to wait for messages expected in subsequent steps. In particular, condition 3. is met. This kind of transformation, which in some cases might require to introduce auxiliary steps, can be applied for every deterministic protocol. Since the transformations are linear, our NP-completeness result for the insecurity of protocols is preserved.

We also point out that the transformations yield more realistic descriptions of protocols in that the actual operations carried out by the principals are formalized. The improved description of A illustrates this. In the original description of A, one implicitly assumes that A can check whether the message expected in A_1 is of the form $Exp(g, x \cdot y)$, i.e., is obtained by exponentiation. However, this is unrealistic since in practice A merely obtains a bit string (which in the second description is z) from which she cannot tell how it was created. The original description of A might therefore exclude possible attacks. We note that the second description of A requires that the exponentiation base may be any message (in the example some message substituted for z). This more realistic version of A could therefore not be formulated when a fixed base is assumed, as is the case in [8].

2.3 The Intruder Model and Attacks

Our intruder model is based on the Dolev-Yao intruder commonly employed in automatic protocol analysis. That is, the intruder has complete control over the network and can derive new messages from his knowledge by composition, decomposition, encryption, and decryption of messages. In addition to the standard Dolev-Yao intruder we equip the intruder with the ability to perform exponentiation (see 4. below).

Given a finite normalized set E of messages, the (infinite) set $forge(E)$ of messages the intruder can derive from E is the smallest set satisfying the following conditions:[1]

1. $E \subseteq forge(E)$.
2. If $\langle m, m' \rangle \in forge(E)$, then $m \in forge(E)$ and $m' \in forge(E)$. Conversely, if $m, m' \in forge(E)$, then $\langle m, m' \rangle \in forge(E)$.
3. If $\{m\}^s_{m'} \in forge(E)$ and $m' \in forge(E)$, then $m \in forge(E)$. Conversely, if $m, m' \in forge(E)$, then $\{m\}^s_{m'} \in forge(E)$. Analogously for asymmetric encryption.
4. If $m, m_1, \ldots, m_n \in forge(E)$, then $\ulcorner Exp(m, m_1^{z_1} \cdots, m_n^{z_n}) \urcorner \in forge(E)$.

[1] In [4] we use an equivalent definition of $forge(E)$ based on derivations.

We call the intruder with these capabilities to derive messages the *DH intruder* where DH stands for Diffie-Hellman. Note that if E is a set of normalized messages, then so is $forge(E)$.

We now define attacks. In an attack on a protocol P, the intruder (nondeterministically) chooses some execution order for P and then tries to produce input messages for the protocol rules. These input messages are derived from the intruder's initial knowledge and the output messages produced by executing the protocol rules. The aim of the intruder is to derive the message secret.

Definition 2. *Let $P = (\{R'_j \Rightarrow S'_j \mid j \in \mathcal{I}\}, <_{\mathcal{I}}, S_0)$ be a protocol. Then an* attack *on P is a tuple (π, σ) where π is an execution ordering on P and σ is a normalized ground substitution of the variables occurring in P such that*

1. *$\ulcorner R_i\sigma \urcorner \in forge(\ulcorner S_0, S_1\sigma, ..., S_{i-1}\sigma \urcorner)$ for every $i \in \{1, \ldots, k\}$ where k is the size of π, $R_i := R'_{\pi^{-1}(i)}$, and $S_i := S'_{\pi^{-1}(i)}$, and*
2. *$\mathsf{secret} \in forge(\ulcorner S_0, S_1\sigma, ..., S_k\sigma \urcorner)$.*

Due to Lemma 1, it does not matter whether, in the above definition, σ is normalized or not. Note that $\ulcorner forge(\ulcorner S_0, S_1\sigma, ..., S_{i-1}\sigma \urcorner)\urcorner = forge(\ulcorner S_0, S_1\sigma, ..., S_{i-1}\sigma \urcorner)$ since from normalized messages the intruder only derives normalized messages.

The decision problem we are interested in is the following set of protocols:

$$\textsc{Insecure} := \{P \mid \text{ there exists an attack on } P\}.$$

Later we will consider minimal attacks.

Definition 3. *Let $P = (\{R_i \Rightarrow S_i,\ i \in \mathcal{I}\}, <_{\mathcal{I}}, S_0)$ be a protocol. An attack (π, σ) is* minimal *if $\Sigma_{x \in \mathcal{V}} |\sigma(x)|_{dag}$ is minimal.*

Clearly, if there is an attack, there is a (not necessarily uniquely determined) minimal attack.

3 Main Theorem and the NP Decision Algorithm

We now state the main theorem of this paper.

Theorem 1. *For the DH intruder, the problem* Insecure *is NP-complete.*

NP-hardness can easily be established (see for instance [1]).

In Figure 1, an NP decision procedure for Insecure is presented where p is a polynomial.[2] It guesses some tuple (π, σ), steps 1. and 2., and then checks whether (π, σ) is an attack on P, steps 3. and 4.

[2] In the technical report we have more fine grained measures for the size of terms and substitutions and more precise bounds. The exact definition of p follows from the proof and is omitted here.

Input: protocol $P = (\{R'_j \Rightarrow S'_j,\ j \in \mathcal{I}\}, <_{\mathcal{I}}, S_0)$

1. Guess an execution ordering π for P. Let k, R_i, and S_i be defined as in Definition 2.
2. Guess a normalized ground substitution σ such that $||\sigma|| \leq p(||P||)$.
3. Test that $\ulcorner R_i\sigma \urcorner \in \mathit{forge}(\ulcorner \{S_j\sigma \mid j < i\} \cup \{S_0\} \urcorner)$ for every $i \in \{1, \ldots, k\}$.
4. Test $\mathsf{secret} \in \mathit{forge}(\ulcorner \{S_j\sigma \mid j < k+1\} \cup \{S_0\} \urcorner)$.
5. If each test is successful, then answer "yes", and otherwise, "no".

Fig. 1. An NP Decision Procedure for INSECURE

Obviously this procedure is sound. It remains to show completeness and it remains to show that the procedure can in fact be carried out in non-deterministic polynomial time. To show the latter, we consider the decision problem

$$\text{DERIVE} := \{(E, m) \mid m \in \mathit{forge}(E)\}$$

where E is a finite set of normalized standard messages and m is a normalized standard message (both given as DAG). In [4], we show:

Proposition 1. *For the DH intruder,* DERIVE *can be decided in deterministic polynomial time.*

Given that the size of σ is polynomially bounded in the size of P, we can conclude from the above proposition that the tests in 3. and 4. of the algorithm can be performed in deterministic polynomial time in the size of P. Thus, the procedure presented in Figure 1 is an NP decision procedure. The basic idea behind the proof of Proposition 1 is to show that i) to derive a message m from E the intruder only needs to derive subterms of E and m in the intermediate steps, and that ii) every derivation step can be carried out in polynomial time.

It remains to show completeness of the procedure in Figure 1. This requires to show that if there exists an attack on P, then there exists an attack (π, σ) on P such that $||\sigma|| \leq p(||P||)$. This is done in two steps: We prove that

1. $|\sigma|_{dag} \leq p'(|P|_{dag})$ for some polynomial p', and
2. $|\sigma|_{exp} \leq p''(||P||)$ for some polynomial p''.

The proof of 1. follows the same lines as the one in [5] where protocols with XOR were considered. However, due to the richer algebraic properties of Diffie-Hellman exponentiation operator, the proof is more involved. Roughly speaking, the main idea of the proof is to show that for a minimal attack (π, σ) on the protocol P if the top symbol of $\sigma(x)$ is an atom, a pair, or encryption, then there exits a $t \in \mathcal{S}(P)$ such that $\ulcorner t\sigma \urcorner = \sigma(x)$. Also, if $\sigma(x)$ is of the form $\mathit{Exp}(t_0, t_1^{z_1} \ldots t_n^{z_n})$, then for every t_i there exists a $t \in \mathcal{S}(P)$ such that $\ulcorner t\sigma \urcorner = \sigma(x)$. In other words, (parts of) σ are build from subterms of P. As an immediate consequence, the number n of factors t_i, $i = 1, \ldots, n$, in products can be bounded in the DAG size of P. More generally, we can prove that the DAG size of σ can be bounded by a polynomial in the DAG size of P (see [4] for details). Note that this does not bound the size of the z_i.

For XOR, to bound the size of σ, only 1. needed to be shown since the number of subterms of σ bounds the size of XOR terms occurring in σ due to the nilpotence of XOR: An XOR term can be considered a product of the form $t_1^{z_1} \dots t_n^{z_n}$ where the z_i are either 0 or 1, and hence, there is no need to bound the z_i's. For the terms considered here this is no longer true, and as mentioned, from 1. we cannot derive a bound on the size of the product exponents z_i. Thus, deciding INSECURE in presence of Diffie-Hellman exponentiation requires new techniques. The main idea behind the proof of 2. is presented in Section 4.

4 Bounding the Product Exponent Size of Attacks

In order to bound $|\sigma|_{exp}$, we will associate with a minimal attack (π, σ) on a protocol P a substitution σ^Z and a (solvable) linear equation system $\mathcal{E}$ such that σ^Z coincides with σ except that the product exponents in σ are replaced by new (integer) variables and such that for every solution β of $\mathcal{E}$, the tuple (π, σ'), where σ' is obtained by replacing in σ^Z the variables according to β, is an attack on P. Since the size of the linear equation system can be bounded polynomially in the size of P, and thus, the size of the minimal solutions of this equation system can be bounded polynomially, we obtain an attack with product exponents polynomially bounded in the size of P (see Proposition 2).

We first need to define messages that may have linear expressions as product exponents.

Definition 4. *Let Z be a set of variables. The set $\mathcal{M} = \mathcal{M}(Z)$ of* open messages *over Z, the set $\mathcal{P} = \mathcal{P}(Z)$ of* open products *over Z, and the set $\mathcal{L}_{exp} = \mathcal{L}_{exp}(Z)$ of* linear expressions *over Z are defined by the following grammar:*

$$\begin{aligned} \mathcal{M} &::= \mathcal{A} \mid \langle \mathcal{M}, \mathcal{M} \rangle \mid \{\mathcal{M}\}^s_{\mathcal{M}} \mid \{\mathcal{M}\}^p_{\mathcal{K}} \mid Exp(\mathcal{M}, \mathcal{P}) \\ \mathcal{P} &::= \mathcal{M}^{\mathcal{L}exp} \mid \mathcal{M}^{\mathcal{L}exp} \cdot \mathcal{P} \\ \mathcal{L}_{exp} &::= \mathbb{Z} \mid Z \mid \mathcal{L}_{exp} + \mathcal{L}_{exp} \mid \mathbb{Z} \cdot \mathcal{L}_{exp} \end{aligned}$$

The size $|e|$ of a linear expression e is the number of characters needed to represent e where integers are encoded in binary. The set of (extended) subterms, the DAG size, and the product exponent size of open messages and products is defined analogously to terms.

We call a mapping $\beta : Z \to \mathbb{Z}$ an *evaluation mapping*. The evaluation $\beta(e) \in \mathbb{Z}$ of a linear expression e w.r.t. β is defined as expected. The evaluation mapping β extends in the obvious way to open messages, open products, sets of open messages, substitutions, etc. A *linear equation system* $\mathcal{E}$ (over Z) is a finite set of equations of the form $e = e'$ where e and e' are linear expressions over Z, e.g., $e = 3 \cdot z - 5 \cdot z'$. The *size* $|\mathcal{E}|$ of $\mathcal{E}$ is $\Sigma_{e=e' \in \mathcal{E}} |e| + |e'|$. An evaluation mapping β is a *solution* of $\mathcal{E}$ ($\beta \models E$) if $\beta(e) = \beta(e')$ for every equation $e = e' \in \mathcal{E}$.

The key for bounding the product exponent size of attacks is the following lemma.

Lemma 2. *Let $t, t_1, \dots, t_n$ be open messages and β be an evaluation mapping such that $\lceil \beta(t) \rceil \in forge(\lceil \beta(t_1) \rceil, \dots, \lceil \beta(t_n) \rceil)$. Then, there exists an extension of β and an equation system $\mathcal{E}$ such that*

1. $\beta \models \mathcal{E}$,
2. $\ulcorner\beta'(t)\urcorner \in \mathit{forge}(\ulcorner\beta'(t_1)\urcorner, \ldots, \ulcorner\beta'(t_n)\urcorner)$ *for every* $\beta' \models \mathcal{E}$, *and*
3. *the size of* $\mathcal{E}$ *is polynomially bounded in* $||t_1, \ldots, t_n, t||$.

The proof of this lemma is quite involved. The basic idea is to replace the messages in a derivation D from $\ulcorner\beta(t_1)\urcorner, \ldots, \ulcorner\beta(t_n)\urcorner$ to $\ulcorner\beta(t)\urcorner$ by open messages which coincide with the messages in D except for the product exponents. More precisely, one first turns the t_i into open messages t'_i, which we call β-normal forms, such that $\beta(t'_i) = \ulcorner\beta(t_i)\urcorner$. Now, one simulates the derivation D in a (symbolic) derivation D' starting with the β-normal forms $t'_1, \ldots, t'_n$. The intermediate terms obtained in D' are β-normal forms of the corresponding terms in D. The equation system $\mathcal{E}$ evolves in the process of turning the t_i into β-normal forms and simulating D.

Using this lemma, it is now rather easy to prove:

Proposition 2. *There exists a polynomial* p'' *such that for every protocol* P, *if* (π, σ) *is a minimal attack on* P, *then there exists an attack* (π, σ') *on* P *such that* σ *and* σ' *coincide up to the product exponents and such that* $|\sigma'|_{exp} \leq p''(||P||)$.

Note that together with the polynomially bound on the DAG size of σ (Section 3), this proposition implies that the size $||\sigma||$ of σ can polynomially be bounded in the size of P. This establishes completeness of the algorithm in Figure 1.

The proof sketch of Proposition 2 is as follows: Let (π, σ) be a minimal attack on P and let σ^Z be σ where all product exponents are replaced by new variables. Let β assign to every of these variables the corresponding product exponent, and thus, $\beta(\sigma^Z) = \sigma$. Then, we apply Lemma 2 to the case where $t = R_i\sigma^Z$ and $t_j = S_j\sigma^Z$ for every $j \in \{0, \ldots, i-1\}$. (W.l.o.g. S_0 can be considered a single message instead of a finite set of messages. We set $R_{k+1} = \mathsf{secret}$.) For every $i \in \{0, \ldots, k+1\}$, Lemma 2 yields an equation system $\mathcal{E}_i$ such that $\beta \models \mathcal{E}_i$ and $\ulcorner\beta'(R_i\sigma^Z)\urcorner \in \mathit{forge}(\ulcorner\beta'(S_0\sigma^Z)\urcorner, \ldots, \ulcorner\beta'(S_{i-1}\sigma^Z)\urcorner$ for every $\beta' \models \mathcal{E}_i$. Let $\mathcal{E}$ be the union of these equation systems. With Lemma 2 and using that the DAG size of σ can polynomially be bounded, it is easy to conclude that the size of $\mathcal{E}$ is polynomially bounded in the size of P. Since $\beta \models \mathcal{E}$, we know that $\mathcal{E}$ is solvable. By [2], there exists a solution β' of $\mathcal{E}$ such that the binary representation of the integers can polynomially be bounded in the size of $\mathcal{E}$, and thus, polynomially bounded in $||P||$. We define σ' to be $\beta'(\sigma^Z)$ and obtain an attack (π, σ') as required.

5 The A-GDH.2 Protocol

We refer the reader to [9] for a detailed description of the A-GDH.2 protocol. Let $\mathrm{P} = \{1, \ldots, n, I\}$ be the set of principals that may be involved in a run of a A-GDH.2 protocol where I is the name of the intruder (who can be both a legitimate participant and a dishonest principal). Any two principals $i, j \in \mathrm{P}$ share a long-term secret key $K_{i,j} (= K_{j,i})$. In a protocol run, a group $G \subseteq \mathrm{P}$ of

principals (membership to a group may vary from one run to another) establish a session key that at the end of the protocol run is only known to the members of the group as long as all members in G are honest (*implicit key authentication*). In a run, one principal plays the role of the so-called *master*. Assume for example that $A, B, C, D \in \mathrm{P}$ want to share a session key and that D is the master. Then, A sends a message to B, B sends a message to C, and C sends a message to the master D. Then, D computes the session key for himself and also broadcasts keying material to A, B, and C using the long-term secret keys shared with these principals who from this material can each derive the session key. We call A the first, B the second, and C the third member of the group.

We now give a formal specification of the protocol in our protocol model. We abbreviate terms $\langle t_1, \langle t_2 \cdots \langle t_{n-1}, t_n \rangle \cdots \rangle\rangle$ by $t_1, \ldots, t_n$. We will define protocol rules $\Pi^{p,j}_{i,l,p'}$ which describe the lth step, $l \in \{1, 2\}$, of principal $p \in \mathrm{P}$, in the jth instance of p, $j \geq 0$, acting as the ith member of the group in which $p' \in \mathrm{P}$ is the master. The relation $\Pi^{p,j}_{i,1,p'} < \Pi^{p,j}_{i,2,p'}$ is the only partial order relationship between protocol rules. By $r^{p,j}$ we denote a random number (an atomic message) generated by p in instance j, and $\mathsf{secret}^{p,j}$ denotes a secret (some atomic message) of p in instance j. We define $\Pi^{p,j}_{1,1,p'}$, i.e. the first step of p in instance j acting as the first member of the group (i.e., the initiator of the protocol) where p' is the master to be

$$1 \Rightarrow \alpha, \mathit{Exp}(\alpha, r^{p,j})$$

where α is a group generator (an atomic message), and for $i > 1$ we define $\Pi^{p,j}_{i,1,p'}$ to be

$$x^{p,j}_1, \ldots, x^{p,j}_i \Rightarrow \mathit{Exp}(x^{p,j}_1, r^{p,j}), \ldots, \mathit{Exp}(x^{p,j}_{i-1}, r^{p,j}), x^{p,j}_i, \mathit{Exp}(x^{p,j}_i, r^{p,j}).$$

where the $x^{p,j}_k$ are variables. The second step $\Pi^{p,j}_{i,2,p'}$ of p in instance j as ith member, $i > 0$, is the protocol rule

$$y^{p,j} \Rightarrow \{\mathsf{secret}^{p,j}\}^s_{\mathit{Exp}(y^{p,j}, r^{p,j} \cdot K^{-1}_{p,p'})}.$$

Note that $\mathit{Exp}(y^{p,j}, r^{p,j} \cdot K^{-1}_{p,p'})$ is the session key computed by p and that implicit key authentication requires that no principal outside of the group can get hold of $\mathsf{secret}^{p,j}$. We now define the protocol rule $M^{p,j}_{p_1 \ldots p_h}$ which describes principal $p \in \mathrm{P}$ in the jth instance acting as master for the group $p_1, \ldots, p_h, p \in \mathrm{P}$ (in this order) where p is the last member of that group. We define $M^{p,j}_{p_1 \cdots p_h}$ to be

$$\begin{aligned} z^{p,j}_1, \ldots, z^{p,j}_{h+1} \Rightarrow\ & \mathit{Exp}(z^{p,j}_1, r^{p,j} \cdot K_{p_1,p}), \ldots, \mathit{Exp}(z^{p,j}_h, r^{p,j} \cdot K_{p_h,p}), \\ & \{\mathsf{secret}^{p,j}\}^s_{\mathit{Exp}(z^{p,j}_{h+1}, r^{p,j})} \end{aligned}$$

where the $z^{p,j}_k$ are variables, $\mathit{Exp}(z^{p,j}_k, r^{p,j} \cdot K_{p_k,p})$ is the keying material for p_k, and the message $\mathit{Exp}(z^{p,j}_{h+1}, r^{p,j})$ is the session key computed by the master p.

The following protocol P describes two sessions of the A-GDH.2 protocol, one for the group $p, p', I, p'' \in \mathrm{P}$ and one for the group p, p', p'' where in both cases p'' is the master of the group. Note that in the first instance, the actions

of the intruder I need not be defined. Formally, the set of protocol rules in P consists of the rules describing p in the first session $\Pi^{p,1}_{1,1,p''}, \Pi^{p,1}_{1,2,p''}$ (note that $\Pi^{p,1}_{1,1,p''} < \Pi^{p,1}_{1,2,p''}$), the rules for p' in the first session $\Pi^{p',1}_{2,1,p''}, \Pi^{p',1}_{2,2,p''}$, and the master $M^{p'',1}_{pp'I}$ of the first session. The protocol rules of the second session are $\Pi^{p,2}_{1,1,p''}, \Pi^{p,2}_{1,2,p''}, \Pi^{p',2}_{2,1,p''}, \Pi^{p',2}_{2,2,p''}, M^{p'',2}_{pp'}$. The initial intruder knowledge is $\{\alpha, r^{I,1}\} \cup \{K_{pI} \mid p \in \mathrm{P}\}$. Let secret be some of the secrets returned by p or p' in the second session. Note that since the intruder is not a member of the group of the second session, he should not be able to obtain secret. However, as shown in [9], there exists an attack on P. It is easy to verify that this attack will be found by our decision procedure.

6 Conclusion

We have shown that a class of protocols with Diffie-Hellman exponents can be automatically analyzed. We strongly conjecture that RSA encryption can easily be integrated into this framework: We have to limit the intruder to only use non-negative product exponents when performing the exponentiation operation and to allow terms of the form t^{-1} (which stands for t's private key) outside of exponents. More generally, in future work we plan to consider the case of arbitrary products outside of exponents and to allow the intruder to perform abelian group operators outside of exponents.

References

1. R. Amadio, D. Lugiez, and V. Vanackere. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, 2002.
2. A. Bockmayr and V. Weispfenning. Solving numerical constraints. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 12, pages 751–842. Elsevier Science, 2001.
3. M. Boreale and M.G. Buscemi. On the symbolic analysis of low-level cryptographic primitives: Modular exponentiation and the Diffie-Hellman protocol. In *Proc. of FCS 2003*.
4. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents. Technical Report 0305, CAU Kiel, Germany, 2003. Available from http://www.informatik.uni-kiel.de/reports/2003/0305.html.
5. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proc. of LICS 2003*.
6. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. of LICS 2003*.
7. C. Meadows and P. Narendran. A unification algorithm for the group Diffie-Hellman protocol. In *Proc. of WITS 2002*.
8. J. Millen and V. Shmatikov. Symbolic protocol analysis with products and Diffie-Hellman exponentiation. In *Proc. of CSFW 16)*, 2003.
9. O. Pereira and J.-J. Quisquater. A Security Analysis of the Cliques Protocols Suites. In *Proc. of CSFW-14*, pages 73–81, 2001.