Complexity Results for Security Protocols with Diffie-Hellman Exponentiation and Commuting Public Key Encryption

YANNICK CHEVALIER
IRIT-Université Paul Sabatier
and
RALF KÜSTERS
ETH Zürich
and
MICHAËL RUSINOWITCH and MATHIEU TURUANI
LORIA-INRIA-Université Henri Poincaré

We show that the insecurity problem for protocols with modular exponentiation and arbitrary products allowed in exponents is NP-complete. This result is based on a protocol and intruder model which is powerful enough to uncover known attacks on the Authenticated Group Diffie-Hellman (A-GDH.2) protocol suite. To prove our results, we develop a general framework in which the Dolev-Yao intruder is extended by generic intruder rules. This framework is also applied to obtain complexity results for protocols with commuting public key encryption.

Categories and Subject Descriptors: []: Computer Security

General Terms: Security Protocols, Verification

Additional Key Words and Phrases: Algebraic Properties, Complexity, Dolev-Yao Model, Diffie-Hellman Exponentiation

1. INTRODUCTION

Designing secure communication systems in open environments, such as the Internet, is a challenging task which heavily relies on cryptographic protocols. However, severe attacks can be conducted on these systems just by exploiting the inherent weaknesses of cryptographic protocols. Attacks on cryptographic protocols are easily overlooked at the design level as adversaries may control the communication network and may combine messages from different protocol sessions. Also, protocol

This paper is a full version of work previously published in [Chevalier et al. 2003a] and [Chevalier et al. 2004]. The authors have partially been supported by PROCOPE and IST AVISPA. The second author was also supported by the DFG.
Author's address: Y. Chevalier, IRIT, 31068 Toulouse, France; R. Küsters, ETH Zürich, Institute of Theoretical Computer Science, Haldeneggsteig 4, CH-8092 Zürich, Switzerland; M. Rusinowitch and M. Turuani, LORIA-INRIA, 54506 Vandoeuvre-les-Nancy, France.
Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
© 20YY ACM 1529-3785/20YY/0700-0001 $5.00

participants may be dishonest. The need for rigorous formal and tool supported analysis of cryptographic protocols has therefore long been realized. The so-called Dolev-Yao model, which has its roots in a paper by Dolev and Yao [Dolev and Yao 1983], is the dominating formal security model in this line of research (see [Meadows 2000] for an overview on the early history on protocol analysis). Many procedures have been proposed to decide security properties of cryptographic protocols in the Dolev-Yao model [Amadio et al. 2002; Boreale 2001; Rusinowitch and Turuani 2001; Millen and Shmatikov 2001] and based on these procedures many tools have been developed (see, e.g., [Millen and Shmatikov 2001; Chevalier and Vigneron 2001; Corin and Etalle 2002]) and successfully been applied to find flaws in published protocols [Clark and Jacob 1997; Boyd and Mathuria 2003; Basin et al. 2003].

Most methods and tools, including those mentioned above, take as a simplifying assumption that the cryptographic algorithms are perfect (*perfect cryptography assumption*). For instance, it is assumed that the decryption key is needed to extract the plain-text from the cipher-text; without such a key, no information whatsoever is leaked about the plain-text. Also, a cipher-text can only be generated with the appropriate key and message. This simple model is insufficient when dealing with the numerous protocols that use operators with algebraic properties, such as the exclusive OR (XOR) and modular exponentiation. The reason for this is twofold: First, without taking these properties into account, the protocols do not even achieve their security goals in the absence of an intruder ([Boyd and Mathuria 2003] contains many examples of such protocols). For instance, the basic Diffie-Hellman key exchange protocol relies on a commutativity property of exponentiation: $(g^a)^b = (g^b)^a$. Second, many attacks exploit algebraic properties of operators, and hence, such attacks cannot be uncovered without considering such properties. For example, the Recursive Authentication Protocol by Bull and Otway [Bull and Otway 1997] was proven to be secure when perfect cryptographic functions are employed [Paulson 1997] and was shown to be insecure when the protocol is implemented using the XOR operator [Ryan and Schneider 1998] and its nilpotency property. Section 6 contains another example, the A-GDH.2 protocol suite (see [Boyd and Mathuria 2003] for more examples). Hence, it is crucial to take algebraic properties of operators such as XOR, Diffie-Hellman exponentiation and RSA encryption into account when analyzing cryptographic protocols.

Contribution of this work. In this paper, we show that the insecurity problem for protocols that use Diffie-Hellman exponentiation with arbitrary products in exponents is NP-complete when the protocols are analyzed w.r.t. a bounded number sessions. We illustrate that our protocol and intruder model is powerful enough to uncover attacks first pointed out by Pereira and Quisquater on the A-GDH.2 protocol suite [Pereira and Quisquater 2001]. The NP-completeness result is also shown for protocols employing commuting public key encryption (such as RSA with common modulus). As a consequence of our proofs, we in addition obtain that the derivation problem, which asks whether the intruder can derive a given message from a given finite set of messages, can be decided in deterministic polynomial time both in case of Diffie-Hellman exponentiation and commuting public key encryption.

The proofs of the NP-completeness results work in two steps: First, we extend the (standard) Dolev-Yao intruder by generic rules, called oracle rules. We show that

the insecurity problem is NP-complete for intruders extended by such rules. Second, we show that the insecurity problem for the intruder extended by the ability to apply Diffie-Hellman exponentiation and commuting public key encryption, respectively, is an instance of the general framework established in the first step.

Related work. The first works to relax the perfect cryptographic assumption by taking algebraic properties of operators into account are [Chevalier et al. 2003a; Comon-Lundh and Shmatikov 2003]. In these works, the Dolev-Yao model was extended by the XOR operator and its algebraic properties. For Diffie-Hellman exponentiation and commuting public key encryption, as studied here, things become, however, much more involved due to the more complex algebraic properties. In particular, unlike the XOR operator, here we do not have the nilpotency property and therefore need to record the number of occurrences of elements in a product, which requires to manipulate equations over integers.

Meadows and Narendran [Meadows and Narendran 2002] designed unification algorithms for handling properties of Diffie-Hellman cryptographic systems. Although these results are useful, they do not solve the more general insecurity problem (see also [Kapur et al. 2003]).

Pereira and Quisquater [Pereira and Quisquater 2004] proposed a systematic way for analyzing protocol suites which extend the Diffie-Hellman key-exchange scheme to a group setting. While they find interesting attacks which exploit algebraic properties of Diffie-Hellman exponentiation, they do not consider decidability questions.

Goubault-Larrecq et al. [Goubault-Larrecq et al. 2005] developed a system to verify protocols using modular exponentiation on a fixed generator g. There method is based on approximations and they have a deduction rule to express that an intruder can generate $g^{a \cdot b}$ from g^a and b, but they do not handle inverses, and hence, they miss realistic attacks.

Boreale and Buscemi [Boreale and Buscemi 2003] addressed a problem similar to ours. However, in their paper, among other restrictions, they put an a priori bound on the number of factors that may occur in products, while in the present paper, we allow an unlimited number of factors. Also, Boreale and Buscemi do not provide a complexity result.

Millen and Shmatikov study Abelian groups and apply them to Diffie-Hellman exponentiation [Millen and Shmatikov 2003], but they do not provide a decision procedure. Also, they assume the base in exponentiations to be a fixed constant. Recently, following the result presented in [Chevalier et al. 2003b], Shmatikov [Shmatikov 2004] proposed a decision procedure for finding attacks in a variant of our model for Diffie-Hellman exponentiation. He does, however, not provide complexity results. While, unlike in our model, in Shmatikov's model products may occur outside of exponents, the factors in products may not start with an exponentiation symbol, a restriction not present in our model.

Structure of the paper. In Section 2, we introduce our protocol and intruder model, including the oracle rules mentioned above. The NP-completeness result for this general framework is proven in Section 3. In Section 4 and 5, we instantiate the oracle rules by rules that allow the intruder to apply Diffie-Hellman exponentiation and show that the derivation problem and the protocol insecurity problem can be

decided in deterministic and non-deterministic polynomial time, respectively. To illustrate our model and results, in Section 6 we formally specify the A-GDH.2 protocol and present an attack on it first discovered by Pereira and Quisquater [Pereira and Quisquater 2001]. Finally, in Section 7 we apply our method to protocols with commuting public key encryption. We conclude in Section 8. Some parts of our proofs are moved to the appendix.

2. THE PROTOCOL AND INTRUDER MODEL

The protocol and intruder model we describe here extends standard models for the (automatic) analysis of security protocols [Amadio et al. 2002; Rusinowitch and Turuani 2001; Millen and Shmatikov 2001] in two respects. First, messages can be built using the operator $Exp(\cdot,\cdot)$, which stands for exponentiation, and a product operator "$\cdot$". Second, in addition to the standard Dolev-Yao rewrite rules, the intruder is equipped with the mentioned oracle rules, which are later instantiated with rules to exponentiate terms. 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:

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

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 $\mathbb{Z}$ 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 symbol $\langle\cdot,\cdot\rangle$ stands for *pairing*, the binary symbol $\{\cdot\}^s_\cdot$ for *symmetric encryption*, and the binary symbol $\{\cdot\}^p_\cdot$ for *public key encryption*. Note that a symmetric key can be any term and that for public key encryption only atomic keys (namely, public and private keys from $\mathcal{K}$) can be used.

The product operator "$\cdot$" models multiplication in an Abelian group. For instance, the product $a^2 \cdot b^3 \cdot c^{-2}$ stands for an element of this group where $a^2 = a \cdot a$, $b^3 = b \cdot b \cdot b$, and $c^{-2} = c^{-1} \cdot c^{-1}$ with c^{-1} the inverse of c. In the A-GDH.2 protocol for example, the Abelian group is a subgroup G of order q of the multiplicative group $\mathbb{Z}_p^*$ where p and q are prime numbers. Terms and products are read modulo commutativity and associativity of the product operator as well as the identity $t^1 = t$. For instance, $d^1 \cdot c^{-2} \cdot (b^3 \cdot a^2)$ and $a^2 \cdot b^3 \cdot c^{-2} \cdot d$ are considered the same products. Also, when we write $t_1^{z_1} \cdots t_n^{z_n}$, then we always assume that the head symbol of t_i is not a product operator. This can always be achieved by resolving parentheses. We write $t_1^* \cdots t_n^*$ to denote a product of the form $t_1^{z_1} \cdots t_n^{z_n}$ where the z_i are some non-zero integers. The operator $Exp(\cdot,\cdot)$ stands for exponentiation. For instance, $Exp(a, b^2 \cdot c^{-1})$ is $a^{b^{2 \cdot c^{-1}}}$.

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*. We often refer

to a term or a product as a "term". We say *standard term* to distinguish a term from a non-standard term.

Variables are denoted by $x, y, \ldots$, terms are denoted by s, t, u, v, finite sets of terms are written $E, F, ...$, and decorations thereof, respectively. We abbreviate $E \cup F$ by E, F, the union $E \cup \{t\}$ by E, t, and $E \setminus \{t\}$ by $E \setminus t$.

For a term t and a set of terms E, $\mathcal{V}(t)$ and $\mathcal{V}(E)$ denote the set of variables occurring in t and E, respectively.

A *ground term* (also called *message*) is a term without variables. We use the expressions *standard* and *non-standard messages* in the same way we use standard and non-standard terms. A *(ground) substitution* σ 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 usual.

We say that two terms t and t' *coincide modulo product exponents* ($t \approx t'$, for short) if t and t' are equal except for the product exponents. For instance:

$$\begin{aligned}\left\langle Exp(g, a^2), Exp(g, b^{-3} \cdot c)\right\rangle &\approx \left\langle Exp(g, a^0), Exp(g, b^2 \cdot c^5)\right\rangle \\ &\not\approx \left\langle Exp(g, b^1), Exp(g, a^2 \cdot c^5)\right\rangle\end{aligned}$$

This equivalence relation extends to substitutions in the obvious way: $\sigma \approx \sigma'$ iff $\sigma(x) \approx \sigma'(x)$ for every variable x.

Given a *standard* term u and a term v, the *replacement* $\delta = [u \leftarrow v]$ of u by v maps every term t to the term $t\delta = t[u \leftarrow v]$ which is obtained by replacing all occurrences of u in t by v. Formally, we have:

Definition 2.1. The application of a replacement $\delta = [u \leftarrow v]$ to a term t, written $t\delta$, is defined inductively on the structure of t as follows:

—If $t = u$, then $t\delta = v$. Otherwise:
—If $t \in \mathcal{A} \cup \mathcal{V}$, then $t\delta = t$.
—If $t = \langle t_1, t_2\rangle$, then $t\delta = \langle t_1\delta, t_2\delta\rangle$.
—If $t = \{t_1\}^s_{t_2}$, then $t\delta = \{t_1\delta\}^s_{t_2\delta}$.
—If $t = \{t_1\}^p_{t_2}$, then $t\delta = \{t_1\delta\}^p_{t_2\delta}$.
—If $t = Exp(t_0, t_1^{z_1} \cdots t_p^{z_p})$, then $t\delta = Exp(t_0\delta, (t_1\delta)^{z_1} \cdots (t_p\delta)^{z_p})$.
—If $t = t_1^{z_1} \cdots t_p^{z_p}$, then $t\delta = (t_1\delta)^{z_1} \cdots (t_p\delta)^{z_p}$.

Note that, for example, with $\delta = [Exp(g, a) \leftarrow 1]$ we have that $Exp(g, a^2)\delta = Exp(g, a^2)$ and $Exp(g, a{\cdot}b)\delta = Exp(g, a{\cdot}b)$. We emphasize that, by definition, $t\delta$ is uniquely determined.

We can compose a substitution σ with a replacement δ: the substitution $\sigma\delta$ maps every $x \in \mathcal{V}$ to $\sigma(x)\delta$.
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 $\mathcal{S}(t) = \{t\} \cup \bigcup_i \mathcal{S}(t_i)$.

Recall that the t_i are standard terms.

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 $\mathcal{S}_{ext}(t) = \mathcal{S}(t) \cup \{M \mid Exp(u, M) \in \mathcal{S}(t)\}$. Thus, the only difference to the definition of subterms is that for subterms $Exp(u, M)$ of t the product M also belongs to the set of (extended) subterms of t.

The set of *factors* of a term t, denoted by $\mathcal{F}(t)$, is recursively defined:

— If t is standard and not $Exp(\cdot, \cdot)$, then $\mathcal{F}(t) = \{t\}$.

— If $t = Exp(u, t_1^* \cdots t_p^*)$, then $\mathcal{F}(t) = \{u, t_1, .., t_p\}$.

— If $t = t_1^* \cdots t_p^*$, then $\mathcal{F}(t) = \{t_1, .., t_p\}$.

Note that $\mathcal{F}(t)$ only contains standard terms. For example, with $a, b, c \in \mathcal{A}$, $\mathcal{F}(a^2 \cdot b^1 \cdot c^{-1}) = \{a, b, c\}$.

We consider two different ways of measuring the size of a term, one includes the product in exponents and the other does not. In any case, the size is defined according to the DAG size of a term. We define $|t| := \mathsf{Card}(\mathcal{S}(t))$ ($|t|_{ext} := \mathsf{Card}(\mathcal{S}_{ext}(t))$) to be the number of (extended) subterms of t.

Remark 2.2. $|t|_{ext} \leq 2 \cdot |t|$.

Note that the definitions of $|\cdot|$ and $|\cdot|_{ext}$ do not measure the size of product exponents. To measure the space needed to represent the product exponents occurring in t, we define

$$||t||_{exp} := \sum_{t_1^{z_1} \cdots t_n^{z_n} \in \mathcal{S}(t)} |z_1| + \ldots + |z_n|,$$

where $|z_i|$ is the number of bits needed to represent the integer z_i in binary. Also,

$$||t|| := |t| + ||t||_{exp}.$$

For a set of terms E the size is defined in the same way (replace t by E in the above definitions). For a substitution σ, we set

$$|\sigma| = \sum_{x \in \mathcal{V}} |\sigma(x)|$$

and analogously, we define $||\sigma||_{exp}$ and $||\sigma||$.

One easily shows by structural induction:

LEMMA 2.3. *Let s be a standard term, t be a term, and x be a variable or an atomic message. Let δ be the replacement $[s \leftarrow x]$. Then, $|t\delta| \leq |t|$.*

We now formulate the algebraic properties of terms. Recall that terms are read modulo commutativity and associativity of the product operator as well as the

identity $t^1 = t$. In addition, we consider the following properties:

$$t \cdot 1 = t \tag{1}$$
$$t^0 = 1 \tag{2}$$
$$1^z = 1 \tag{3}$$
$$t^z \cdot t^{z'} = t^{z+z'} \tag{4}$$
$$Exp(t, 1) = t \tag{5}$$
$$Exp(Exp(t, t'), t'') = Exp(t, t' \cdot t'') \tag{6}$$

A *normal form* of a term is obtained by iteratively applying these identities from left to write. Note that the identities can be applied to subterms of terms and that the normal forms are uniquely determined up to commutativity and associativity of the product operator. In other words, the rewriting system induced by the above identities when oriented from left to right is confluent modulo commutativity and associativity of the product operator. Since we consider terms modulo commutativity and associativity of the product operator, normal forms are uniquely determined. The normal form of a term t is denoted by $\ulcorner t \urcorner$. We illustrate the notion of a normal form by some examples: If $a, b, c, d \in \mathcal{A}$, then

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

The normal forms of sets of terms and substitutions are defined in the obvious way. A term t is *normalized* if $\ulcorner t \urcorner = t$. In the same way normalized sets and substitutions are defined. Two terms t and t' are *equivalent* (modulo *Exp* and $\cdot$) if $\ulcorner t \urcorner = \ulcorner t' \urcorner$. One easily shows:

LEMMA 2.4. *For every term t, t' and substitution σ:*

(1) $\mathcal{S}(\ulcorner t \urcorner) \subseteq \ulcorner \mathcal{S}(t) \urcorner$,
(2) $||\ulcorner t \urcorner||_{exp} \leq ||t||_{exp}$,
(3) $||\ulcorner t \urcorner|| \leq ||t||$,
(4) $\mathcal{S}(t\sigma) \subseteq \mathcal{S}(t)\sigma \cup \mathcal{S}(\mathcal{V}(t)\sigma)$, *and*
(5) $\ulcorner t\sigma \urcorner = \ulcorner \ulcorner t \urcorner \sigma \urcorner = \ulcorner t \ulcorner \sigma \urcorner \urcorner = \ulcorner \ulcorner t \urcorner \ulcorner \sigma \urcorner \urcorner$.

2.2 The Intruder Model

Our intruder model follows the Dolev-Yao intruder [Dolev and Yao 1983]. That is, the intruder has complete control over the network and he can derive new messages from his initial knowledge and the messages received from honest principals during protocol runs. To derive a new message, the intruder can compose and decompose, encrypt and decrypt messages, in case he knows the key. What distinguishes the intruder we consider here from the standard Dolev-Yao intruder is that we equip the intruder with guess rules which provide him with additional capabilities for deriving messages. In Section 2.5, we consider guess rules that satisfies certain conditions. We will call these rules oracle rules.

	Decomposition rules	Composition rules
Pair	$L_{p1}(\langle m, m'\rangle)$: $\langle m, m'\rangle \to m$ $L_{p2}(\langle m, m'\rangle)$: $\langle m, m'\rangle \to m'$	$L_c(\langle m, m'\rangle)$: $m, m' \to \langle m, m'\rangle$
Asymmetric	$L_{ad}(\{m\}^p_K)$: $\{m\}^p_K, K^{-1} \to m$	$L_c(\{m\}^p_K)$: $m, K \to \{m\}^p_K$
Symmetric	$L_{sd}(\{m\}^s_{m'})$: $\{m\}^s_{m'}, m' \to m$	$L_c(\{m\}^s_{m'})$: $m, m' \to \{m\}^s_{m'}$
Guess	$L_{od}(m)$: $E \to m$ with m subterm of E and E normalized.	$L_{oc}(m)$: $E \to m$ with E, m normalized and such that every proper subterm of m is a subterm of E.

Table I. Intruder rules.

The intruder derives new messages from a given (finite) set of message by applying intruder rules. An *intruder rule* (or *t-rule*) L is of the form $S \to t$, where S is a finite set of *standard* messages and t is a *standard* message. Given a finite set E of standard messages, the rule L *can be applied* to E if $S \subseteq E$. We define the *step relation* $\to_L$ induced by L to be a binary relation on finite sets of standard messages. For every finite set of standard messages E we have $E \to_L E, t$ (recall that E, t stands for $E \cup \{t\}$) if L is a t-rule and L can be applied to E. If $\mathcal{L}$ denotes a (finite or infinite) set of intruder rules, then $\to_{\mathcal{L}}$ denotes the union $\bigcup_{L\in\mathcal{L}} \to_L$ of the step relations $\to_L$ with $L \in \mathcal{L}$. With $\to^*_{\mathcal{L}}$ we denote the reflexive and transitive closure of $\to_{\mathcal{L}}$.

The set of intruder rules we consider in this paper is depicted in Table I. In this table, m, m' denote arbitrary standard messages, K is an element of $\mathcal{K}$, and E is a finite set of standard messages.

We emphasize that the notion of *intruder rule* will always refer to the rules listed in Table I. For now, there may be any set of guess rules of the kind shown in Table I, later we will consider certain classes of guess rules, namely oracle rules.

The intruder rules are denoted as shown in Table I. With $L_{od}(m)$ and $L_{oc}(m)$ we denote (finite or infinite) sets of guess rules. For uniformity, we therefore consider $L_{p1}(\langle m, m'\rangle), \ldots, L_{sd}(\{m\}^s_{m'})$ and $L_c(\langle m, m'\rangle), \ldots, L_c(\{m\}^s_{m'})$ as singletons. Note that, even if there are no guess rules, the number of decomposition and composition rules is always infinite since there are infinitely many messages m, m'.

We further group the intruder rules as follows. In the following, t ranges over all standard messages.

— $L_d(t) := L_{p1}(t) \cup L_{p2}(t) \cup L_{ad}(t) \cup L_{sd}(t)$. In case, for instance, $L_{p1}(t)$ is not defined, i.e., the head symbol of t is not a pair, then $L_{p1}(t) = \emptyset$; analogously for the other rule sets,
— $L_d := \bigcup_t L_d(t)$, $L_c := \bigcup_t L_c(t)$,
— $L_{od} := \bigcup_t L_{od}(t)$, $L_{oc} := \bigcup_t L_{oc}(t)$,
— $L_o(t) := L_{oc}(t) \cup L_{od}(t)$, $L_o := L_{oc} \cup L_{od}$,
— $\mathcal{L}_d := \bigcup_t \mathcal{L}_d(t)$ where $\mathcal{L}_d(t)$ is the set of all decomposition t-rules in Table I, i.e., all t-rule in the left column of the table,
— $\mathcal{L}_c := \bigcup_t \mathcal{L}_c(t)$ where $\mathcal{L}_c(t)$ is the set of all composition t-rules in Table I, and
— $\mathcal{L} := \mathcal{L}_d \cup \mathcal{L}_c$.

Note that $\mathcal{L}$ denotes the (infinite) set of all intruder rules we consider here. The set of messages the intruder can derive from a (finite) set E of messages is:

$$\mathit{forge}(E) := \bigcup\{E' \mid E \to_{\mathcal{L}}^{*} E'\}.$$

From the definition of intruder rules in Table I it immediately follows:

LEMMA 2.5. *If E is a normalized set of messages, then $\mathit{forge}(E)$ is normalized.*

The lemma says that if an intruder only sees normalized messages, then he only creates normalized messages. Intruders should be modeled in such a way that they cannot distinguish between equivalent messages. In what follows we always assume that the intruder's knowledge consists of a set of normalized messages, where every single normalized message in this set can be seen as a representative of its equivalence class.

2.3 Protocols

Informally speaking, a protocol consists of a finite set of principals and every principal performs a finite and fixed sequence of receive-send actions. Since, as usual in the Dolev-Yao model, we assume that the intruder controls the communication network, all messages sent by a principal are sent to the intruder and all messages received by a principal come from the intruder. When a principal receives a message from the intruder, he performs his current receive-send action, i.e., after receipt of the message and some computation a message is sent back to the intruder (if any). The next message received from the intruder is processed by the next receive-send action in the sequence in the same way.

Similar to other models [Rusinowitch and Turuani 2001; Millen and Shmatikov 2001; Chevalier et al. 2003a], we model receive-send actions by rewrite rules of the form $R \Rightarrow S$. On receiving a message m, it is first checked whether m and R match, i.e., whether there exists a ground substitution σ such that $\ulcorner m \urcorner = \ulcorner R\sigma \urcorner$. If so, $\ulcorner S\sigma \urcorner$ is returned as output. We always assume that the messages exchanged between the principals and the intruder are normalized. In particular, m is assumed to be normalized and the output of the above rule is not $S\sigma$ but $\ulcorner S\sigma \urcorner$. 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). We also note that since the different protocol rules in the specification of a protocol may share variables, some of the variables in R and S may be bounded already by substitutions obtained from applications of previous protocol rules.

Instead of defining principals as a sequence of rewrite rules of the above form and protocols as a finite set of principals, our definition is slightly more general. We will not define principals explicitly but define protocols as a finite partially ordered set of rewrite rules. Such a partial ordering can contain several linear orderings of rewrite rules corresponding to sequences of rewrite rules, and hence, principals. Before defining protocols formally, let us take a look at an example: The set of

rules of the example protocol is defined to be

$$\begin{array}{lrcl} 1: & \mathit{Start} & \Rightarrow & N_a \\ 2: & \{\langle N_a, x\rangle\}_K^s & \Rightarrow & \mathit{End} \\ 1': & y & \Rightarrow & \{\langle y, N_b\rangle\}_K^s \end{array}$$

with the partial ordering $1 < 2$. Intuitively, rules 1 and 2 represent one principal, say Alice, and step $1'$ represents another principal, say Bob. The partial ordering guarantees that Alice performs 1 before 2, but $1'$ can be performed before 1, between 1 and 2, or after 2. The atom N_a is a nonce generated by Alice, N_b is a nonce generated by Bob, and K is the key shared between Alice and Bob. It is assumed that the atom Start belongs to the initial intruder knowledge.

The formal definition of protocols and rewrite rules is as follows. The definition of protocols also contains the initial intruder knowledge as we are interested in attacks on protocols (Section 2.4). The three conditions required of protocols are explained following the definition.

Definition 2.6. 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 the following conditions are satisfied:

(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 $x \in \mathcal{V}(R_j)$.
(3) For every subterm $\mathit{Exp}(t_1, t_2^{z_2} \cdots t_n^{z_n})$ of R_i, there exists $k \in \{1, .., n\}$ such that

$$\mathcal{V}(t_l) \subseteq \bigcup_{j <_{\mathcal{I}} i} \mathcal{V}(R_j) \quad \text{for every} \quad l \in \{1, .., n\} \setminus \{k\}.$$

Given a protocol P, in the following we let $\mathcal{A}$ denote the set of constants occurring in P. We define $\mathcal{S}(P) := E \cup \bigcup_{i \in \mathcal{I}} (R_i \cup S_i)$ to be the *set of subterms of* P, $\mathcal{V} := \mathcal{V}(P)$ to be the set of variables occurring in P, and $|P| := |\mathcal{S}(P)|$, $||P|| := ||\mathcal{S}(P)||$, $|P|_{ext} := |\mathcal{S}(P)|_{ext}$, and $||P||_{ext} := ||\mathcal{S}(P)||_{ext}$ to be the different *sizes of* P.

Condition 1. is w.l.o.g., since due to Lemma 2.4, the transformation performed by a protocol rule and its normalized variant coincide.

Condition 2. guarantees that when with S_i an output is produced, all variables in S_i are "bounded" already, i.e., the substitution of these variables is determined by the previously received messages from the intruder. Otherwise, the output of a protocol rule would be arbitrary since unbounded variables could be mapped to any message.

Condition 3. guarantees that all exponents, except for at most one exponent t_k, in any subterm of R_i of the form $\mathit{Exp}(\cdot, \cdot)$ are built over variables from previous steps, i.e. variables that have been assigned messages before the ith step. Below we argue that in order to not miss attacks, protocol specifications should not contain variables in exponents whose values are not determined by previous steps. Hence, Condition 3. could be restricted even more by requiring that $\mathcal{V}(t_l) \subseteq \bigcup_{j <_{\mathcal{I}} i} \mathcal{V}(R_j)$ also for $l = k$. However, we use (3) in Definition 2.6 as it is sufficient and convenient to use in proofs.

Specification of Protocols Involving Diffie-Hellman Exponentiation—Discussion. Typically, protocols that employ Diffie-Hellman exponentiation require to check whether a given message belongs to a certain group (see, e.g., [Steiner et al. 1998]), i.e., given a group generator g and a message m, one checks whether there exists a message (number) a such that $m = g^a$. It is tempting to model this by a term of the form $Exp(g, x)$ where g is a constant which stands for the group generator g and x is a variable. In the symbolic world, such a term would only match with messages of the form $Exp(g, m')$ for some m', and hence, with messages guaranteed to belong to the group generated by g. However, a constant b (when interpreted as a bit string) or the pair $\langle c, d\rangle$ with constants c and d (when interpreted as the concatenation of two bit strings c and d) might also represent elements of the group generated by g. Thus, if the adversary sends $Exp(b, m')$ or $Exp(\langle c, d\rangle, m')$ for some message m', then in the cryptographic world both messages would be accepted, i.e., pass the group membership test. However, in the symbolic world these messages would not match with $Exp(g, x)$. As a consequence, in the symbolic world one might miss attacks that are possible in the cryptographic world. In order to prevent this, we suggest to use in protocol specifications variables instead of terms of the form $Exp(g, x)$ or more generally $Exp(g, t)$ where t is some term containing a variable. That is, instead of writing, for example, $Exp(g, x) \Rightarrow \ldots$ we suggest to write $z \Rightarrow \ldots$ for some variable z. In such a specification terms of the form, for example, $Exp(b, m')$ or $Exp(\langle c, d\rangle, m')$ would be accepted. Of course, messages could be accepted that w.r.t. the cryptographic interpretation would not belong to the group generated by g, i.e., false attacks could occur. However, it seems quite likely that, in a symbolic model, if there exists an attack on a protocol modeled as just described, then in fact something is wrong with the protocol independent of the issue of membership checks. Finally, we note that our modeling avoids the need for instantiating variables by non-standard terms since messages of the form $Exp(m, m')$ only need to be matched with variables.

2.4 Attacks

We now define attacks on protocols. We first need to introduce the notion of an execution ordering of a protocol: a linear ordering of some of the protocol rules in a protocol consistent with the partial ordering.

Definition 2.7. A bijective mapping $\pi : \mathcal{I}' \to \{1, .., p\}$ is called *execution ordering* for P if $\mathcal{I}' \subseteq \mathcal{I}$, p 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 p.

Informally speaking, in an attack on a protocol P, the intruder chooses some execution order for P and then tries to produce input messages for the protocol rules in the order determined by the execution ordering. These input messages are derived from the intruder's initial knowledge and the output messages received so far. The aim of the intruder is to derive the message secret. If different sessions of a protocol running interleaved shall be analyzed, then these sessions must be encoded into the protocol P. This is the standard approach when protocols are analyzed w.r.t. a bounded number of sessions, see, for instance, [Rusinowitch and Turuani 2001; Millen and Shmatikov 2001].

Definition 2.8. Let $P = (\{R'_\iota \Rightarrow S'_\iota \mid \iota \in \mathcal{I}\}, <_\mathcal{I}, S_0)$ 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) $\ulcorner R_i\sigma \urcorner \in \mathit{forge}(\ulcorner S_0, S_1\sigma, ..., S_{i-1}\sigma \urcorner)$ 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) $\mathsf{secret} \in \mathit{forge}(\ulcorner S_0, S_1\sigma, ..., S_k\sigma \urcorner)$.

Due to Lemma 2.4, it does not matter whether, in the above definition, σ is normalized or not. Also note that Lemma 2.5 implies: $\ulcorner \mathit{forge}(\ulcorner S_0, S_1\sigma, ..., S_{i-1}\sigma \urcorner) \urcorner = \mathit{forge}(\ulcorner S_0, S_1\sigma, ..., S_{i-1}\sigma \urcorner)$.

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 2.9. Let $P = (\{R_\iota \Rightarrow S_\iota \mid \iota \in \mathcal{I}\}, <_\mathcal{I}, S_0)$ be a protocol. An attack (π, σ) is *minimal* if $|\sigma|$ is minimal, i.e., for all substitutions σ', if (π, σ') is an attack on P, then $|\sigma| \leq |\sigma'|$.

Clearly, if there is an attack, there is a minimal attack. Note, however, that minimal attacks are not necessarily uniquely determined.

2.5 Oracle Rules

Oracle rules are guess rules which satisfy certain conditions. To define these rules, we first need some new notions.

A *derivation* D of length n, $n \geq 0$, is a sequence of steps of the form $E \rightarrow_{L_1} E, t_1 \rightarrow_{L_2} \cdots \rightarrow_{L_n} E, t_1, \ldots, t_n$ with a finite set of standard messages E, standard messages $t_1, \ldots, t_n$, and intruder rules $L_i \in \mathcal{L}$ such that $E, t_1, \ldots, t_{i-1} \rightarrow_{L_i} E, t_1, \ldots, t_i$ and $t_i \notin E \cup \{t_1, \ldots, t_{i-1}\}$ for every $i \in \{1, \ldots, n\}$. The rule L_i is called the *ith rule* of D and the step $E, t_1, \ldots, t_{i-1} \rightarrow_{L_i} E, t_1, \ldots, t_i$ is called the *ith step* of D. We write $L \in D$ to say that $L \in \{L_1, \ldots, L_n\}$. If S is a set of intruder rules, then we write $S \notin D$ to say $S \cap \{L_1, \ldots, L_n\} = \emptyset$. The message t_n is called the *goal* of D.

We also need *well-formed* derivations, which are derivations where every message generated by an intermediate step either occurs as subterm in the goal or in the initial set of messages.

Definition 2.10. Let $D = E \rightarrow_{L_1} \ldots \rightarrow_{L_n} E'$ be a derivation with goal t. Then, D is *well-formed* if $E' \subseteq \mathcal{S}(E, t)$.

We can now define oracle rules. Condition 1. in the following definition will allow us to bound the length of derivations. The remaining conditions allow us to replace a subterm u in σ by a smaller message and are later used to bound the size of the substitution σ in an attack.

Definition 2.11. Let $L_o = L_{oc} \cup L_{od}$ be a (finite or infinite) set of guess rules, where L_{oc} and L_{od} denote disjoint sets of composition and decomposition guess rules, respectively. Then, L_o is a *set of oracle rules* (w.r.t. $L_c \cup L_d$ as defined above) iff all of the following conditions are satisfied:

Input: protocol $P = (\{R_\iota \Rightarrow S_\iota,\ \iota \in \mathcal{I}\}, <_\mathcal{I}, S_0)$. Recall that $\mathcal{V} = \mathcal{V}(P)$.

(1) Guess an execution ordering π for P. Let k, R_i, and S_i be defined as in Definition 2.8.
(2) Guess a normalized ground substitution σ such that $|\mathcal{V}\sigma| \leq |P|$ and $||\sigma||_{exp} \leq p(||P||)$.
(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. NP Decision Procedure for INSECURE where p denotes the polynomial bounding the size of product exponents.

(1) For every message t and finite set E, if $t \in \mathit{forge}(E)$, then there exists a well-formed derivation from E with goal t.
(2) If $F \rightarrow_{L_{oc}(t)} F, t$ and $F, t \rightarrow_{L_d(t)} F, t, a$, then there exists a derivation D from F with goal a such that $L_d(t) \notin D$.
(3) For every finite set F of messages with $1 \in F$, if $F \setminus u \rightarrow_{\mathcal{L}_c(u)} F$, i.e., u can be composed from $F \setminus u$ in one step, then $F \rightarrow_{L_o(t)} F, t$ implies $\ulcorner t[u \leftarrow 1] \urcorner \in \mathit{forge}(\ulcorner F[u \leftarrow 1] \urcorner)$ for every message t.

3. THE NP DECISION ALGORITHM

We now state one of the main theorems of this paper, which says that INSECURE is decidable in non-deterministic polynomial time for every set of oracle rules satisfying certain conditions. This generic theorem is then applied in Section 4 and Section 7 to show that INSECURE is in NP in presence of an intruder that can perform Diffie-Hellman exponentiation and exploit commutative public-key encryption, respectively.

In the theorem, two conditions are required of the set of oracle rules. The first is that the oracle rule problem can be decided efficiently and the second is that the set of oracle rules allows polynomial product exponent attacks.

The *oracle rule problem* is defined as follows:

$$\text{ORACLERULE} = \{(E, m) \mid E \rightarrow_{L_o} E, m\}$$

where E is a finite set of standard messages and m is a standard message, both given as DAGs.

We say that the set of oracle rules L_o allows *polynomial product exponent attacks* if for every protocol P and every minimal attack (π, σ) on this protocol there exists σ' such that $\sigma' \approx \sigma$ (recall that this means that σ' and σ coincide modulo the product exponents), $(\pi, \ulcorner \sigma' \urcorner)$ is an attack on P, and $||\sigma'||_{exp}$ is polynomially bounded in $||P||$. Note that by Lemma 2.4 this implies that $||\ulcorner \sigma' \urcorner||_{exp}$ is polynomially bounded in $||P||$ as well.

THEOREM 3.1. *Let L_o be a set of oracle rules. If*

— ORACLERULE $\in$ *PTIME and*
— *L_o allows polynomial product exponent attacks,*

then INSECURE *is in NP.*

The theorem is proved by showing that the non-deterministic algorithm depicted in Figure 1 is a non-deterministic polynomial-time algorithm and that it decides INSECURE. The polynomial p, which is a polynomial bounding the size of product exponents, exists due to the assumption that L_o allows polynomial product exponent attacks. The structure of the algorithm is as follows: In step 1. and 2., a "small" attack (π, σ) on the given protocol P is guessed. Then, in step 3. and 4. it is checked whether (π, σ) is in fact an attack on P.

Obviously, the algorithm in Figure 1 is sound. The main problem is to show that it is complete and that it runs in non-deterministic polynomial-time. This is shown using results proved in Sections 3.1, 3.2 and 3.3. In what follows, we first explain the results proved in these sections and then using these results prove completeness of the algorithm and the claimed complexity.

In Section 3.1, we show (see Theorem 3.2) that the following problem, henceforth called *derivation problem*, can be solved in polynomial time in $||E, t||$, provided that ORACLERULE can be decided in deterministic polynomial time:

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

where E is a finite set of standard messages and t is a standard message, given as DAGs.

In Section 3.2, we show (see Proposition 3.14) that substitutions of minimal attacks are built from subterms of terms occurring in the description of the protocol.

Using Proposition 3.14, in Section 3.3 we bound the number of subterms in substitutions of minimal attacks and obtain that $|\mathcal{V}\sigma| \leq |P|$ (Corollary 3.16).

Given these results, we can now prove Theorem 3.1.

Proof of Theorem 3.1. We show that the algorithm depicted in Figure 1 runs in non-deterministic polynomial time and that it is sound and complete. Clearly, as already mentioned, the algorithm is sound.

To show completeness, we need to prove that if there exists an attack (π, σ) on P, then there is one with the size of σ bounded as in step 2. of the algorithm in Figure 1. This immediately follows from Corollary 3.16 and our assumption that L_o allows polynomial product exponent attacks.

It remains to show that our algorithm runs in non-deterministic polynomial time in the size $||P||$ of P. Clearly, this is the case of step 1. and 2. of our algorithm. To show this for step 3. and 4. we use Theorem 3.2. Let E be the set $\ulcorner\{S_j\sigma \mid j < i\} \cup S_0\urcorner$ for some $i \in \{1, \ldots, k\}$ and t be $\ulcorner R_i\sigma\urcorner$ or secret. By Theorem 3.2, we can decide if $t \in \mathit{forge}(E)$ in deterministic polynomial time in $||E, t||$. Corollary 3.16 implies that $|E, t| \leq |P|$. Since $||\sigma||_{exp}$ is polynomially bounded in $||P||$, this is also the case for $||\{S_j\sigma \mid j < i\} \cup S_0 \cup \{R_i\sigma\}||_{exp}$. By Lemma 2.4, it follows that $||E, t||_{exp}$ is polynomially bounded in $||P||$. Consequently, step 3. and 4. can be carried out in deterministic polynomial-time in $||P||$. □

3.1 Deciding the Derivation Problem

We show that the derivation problem can be decided in polynomial time given that this is the case for the problem ORACLERULE.

THEOREM 3.2. DERIVE $\in$ *PTIME provided that* ORACLERULE $\in$ *PTIME.*

PROOF. Let $d_t(E)$ be the set consisting of the messages $t' \in \mathcal{S}(E,t)$ that can be derived from E in one step. Using that the number of terms $t' \in \mathcal{S}(E,t)$ is linear in $||E,t||$ and that $E \rightarrow_{L_o} E,t$ can be checked in polynomial time it is easy to see that $d_t(E)$ can be computed in polynomial time in $||E,t||$. Now, if $t \in \mathit{forge}(E)$, then Definition 2.11 guarantees that there exists a well-formed derivation $D = E \rightarrow_{L_1} E,t_1 \rightarrow \ldots \rightarrow_{L_r} E,t_1,..,t_r$, with $t_r = t$. In particular, $t_i \in \mathcal{S}(E,t)$ for every $i \in \{1,\ldots,k\}$. By definition of derivations, all t_i are different. It follows $r \leq |t,E|$. Moreover, with $d_t^0(E) := E$ and $d_t^{l+1}(E) := d_t(d_t^l(E))$ we have that $t \in d_t^{|E,t|}(E)$ iff $t \in \mathit{forge}(E)$. Since $d_t^{|E,t|}(E)$ can be computed in polynomial time, the theorem follows. □

3.2 Characterizing the Subterms of Minimal Attacks

We now show that substitutions of minimal attacks can be constructed by linking subterms that are initially occurring in the problem specification (Proposition 3.14).

In order to prove the proposition, we first show some properties of replacements (Lemmas 3.4 to 3.6), properties of substitutions in minimal attacks (Lemmas 3.7 to 3.9), and properties of derivations (Lemmas 3.10 to 3.13).

In what follows, we assume that L_o is a set of oracle rules. If $t \in \mathit{forge}(E)$, we denote by $D_t(E)$ a well-formed derivation from E with goal t (chosen arbitrarily among the possible ones). Note that there always exists such a derivation due to the definition of oracle rules.

Let $P = (\{R_\iota \Rightarrow S_\iota,\ \iota \in \mathcal{I}\}, <_\mathcal{I}, S_0)$ be a protocol and (π,σ) be an attack on P. Let k, R_i, and S_i be defined as in Definition 2.8. Recall that $\mathcal{S}(P)$ is the set of subterms of P, $\mathcal{A} \subseteq \mathcal{S}(P)$, and $\mathcal{V} = \mathcal{V}(P)$ is the set of variables occurring in the protocol. Also, we need the following crucial notion.

Definition 3.3. Let t and t' be two terms and θ a ground substitution. Then, t is a *θ-match* of t', denoted $t \sqsubseteq_\theta t'$, if t and t' are standard and $\ulcorner t\theta \urcorner = t'$.

3.2.1 *Properties of Replacements.* The following lemmas state distributivity properties between the normalization function, substitutions, the exponentiation operator, and replacements.

LEMMA 3.4. *Let u be a normalized term, M, M' be two products such that for all $t \in \mathcal{F}(M) \cup \mathcal{F}(M')$, t is normalized. Let s be a standard normalized term and δ the replacement $[s \leftarrow 1]$. Then:*

(1) $\ulcorner (M \cdot M')\delta \urcorner = \ulcorner M \cdot M' \urcorner \delta \urcorner$, *in particular,* $\ulcorner M\delta \urcorner = \ulcorner M \urcorner \delta \urcorner$.

(2) $\ulcorner \mathit{Exp}(u,M)\delta \urcorner = \ulcorner \mathit{Exp}(u,M) \urcorner \delta \urcorner$ *if* $s \neq \ulcorner \mathit{Exp}(u,M) \urcorner$ *and, in case s is of the form $\mathit{Exp}(\cdot,\cdot)$, also $s \neq u$.*

PROOF. See Appendix 9.1. □

We note that 2. in the previous lemma does not hold without the restrictions on s. The following example shows the problem in case $s = \ulcorner \mathit{Exp}(u,M) \urcorner$: Assume that $s = \mathit{Exp}(a,b)$, $u = a$ and $M = b \cdot c \cdot c^{-1}$. Then, $s = \ulcorner \mathit{Exp}(u,M)\delta \urcorner \neq \ulcorner \mathit{Exp}(u,M) \urcorner \delta \urcorner = 1$. The next example illustrates why $s \neq u$ is necessary: Define $s = u = \mathit{Exp}(a,b)$ and $M = c$. Then, $1 = \ulcorner \mathit{Exp}(u,M)\delta \urcorner \neq \ulcorner \mathit{Exp}(u,M) \urcorner \delta \urcorner = \mathit{Exp}(a, b \cdot c)$.

LEMMA 3.5. *Let σ be a normalized ground substitution, E a set of normalized terms, s a normalized standard non atomic term, and δ the replacement $[s \leftarrow 1]$. Let $\sigma' = \ulcorner\sigma\delta\urcorner$. If there is no standard subterm t of E such that $t \sqsubseteq_\sigma s$, then $\ulcorner E\sigma'\urcorner = \ulcorner\ulcorner E\sigma\urcorner\delta\urcorner$.*

PROOF. See Appendix 9.1. □

LEMMA 3.6. *Let $t', t_1, \ldots, t_n, t, u$ be normalized standard terms, $z_1 \ldots, z_n \in \mathbb{Z}$, and let δ be the replacement $[u \leftarrow 1]$ such that $u \neq t$, and $t = \ulcorner Exp(t', t_1^{z_1} \cdots t_n^{z_n})\urcorner$. If $t' = Exp(\cdot,\cdot)$, then we also assume that $u \neq t'$. Then,*

$$\ulcorner t\delta\urcorner = \ulcorner Exp(\ulcorner t'\delta\urcorner, \ulcorner t_1\delta\urcorner^{z_1} \cdots \ulcorner t_n\delta\urcorner^{z_n})\urcorner.$$

PROOF. See Appendix 9.1 □

3.2.2 *Properties of Substitutions of Minimal Attacks.* The following lemma allows to prove what we will call the unique matching property.

LEMMA 3.7. *Let s be a standard term, t be a normalized term, and σ be a normalized substitution such that $s \in \mathcal{S}(\ulcorner t\sigma\urcorner)$ and $s \notin \mathcal{S}(x\sigma)$ for every $x \in \mathcal{V}(t)$. Then, there exists a standard subterm t' of t with $t' \sqsubseteq_\sigma s$.*

PROOF. By Lemma 2.4 we have

$$\begin{aligned}\mathcal{S}(\ulcorner t\sigma\urcorner) &\subseteq \ulcorner\mathcal{S}(t\sigma)\urcorner \\ &\subseteq \ulcorner\mathcal{S}(t)\sigma \cup \mathcal{V}(t)\sigma\urcorner.\end{aligned}$$

Since σ is in normal form this implies

$$\mathcal{S}(\ulcorner t\sigma\urcorner) \subseteq \ulcorner\mathcal{S}(t)\sigma\urcorner \cup \mathcal{V}(t)\sigma.$$

By assumption, $s \in \mathcal{S}(\ulcorner t\sigma\urcorner)$ and $s \notin \mathcal{S}(\mathcal{V}(t)\sigma)$. It follows, $s \in \ulcorner\mathcal{S}(t)\sigma\urcorner$, which means that there exists $t' \in \mathcal{S}(t)$ such that $\ulcorner t'\sigma\urcorner = s$. □

We now prove the *unique matching property*, which says that if the intruder delivers messages $m_1, \ldots, m_i$, then there is only at most one way to match these matches against $R_1, \ldots, R_i$ with R_j, $j \in \{1, \ldots, i\}$, defined as above.

LEMMA 3.8. *Given any sequence of normalized messages $m_1, \ldots, m_i$, there exists at most one normalized substitution σ such that $\ulcorner R_j\sigma\urcorner = m_j$ for $j \in \{1, \ldots, i\}$.*

PROOF. Assume by contradiction that the lemma does not hold and let $i \in \{1, \ldots, n\}$ be minimal such that there exist two different normalized substitutions σ and σ' such that $\ulcorner R_j\sigma\urcorner = \ulcorner R_j\sigma'\urcorner$ for $j \in \{1, \ldots, i\}$. By minimality of i the substitutions σ and σ' coincide on $V_{i-1} = \mathcal{V}(R_1, \ldots, R_{i-1})$ and they differ on a variable in $\mathcal{V}(R_i) \setminus V_{i-1}$. Let σ_0 be the substitution equal to σ on V_{i-1} and equal to the identity on $\mathcal{V}(R_i) \setminus V_{i-1}$, and let $r = \ulcorner R_i\sigma_0\urcorner$.

By Lemma 2.4, 5. and since $R_i\sigma = (R_i\sigma_0)\sigma$ and $R_i\sigma' = (R_i\sigma_0)\sigma'$ we have that $\ulcorner R_i\sigma\urcorner = \ulcorner r\sigma\urcorner = \ulcorner r\sigma'\urcorner$. By assumption, there exists $x \in \mathcal{V}(r)$ such that $x\sigma \neq x\sigma'$. Let $t_x \in \mathcal{S}(r)$ be minimal w.r.t. the subterm relation such that $x \in \mathcal{V}(t_x)$ and $\ulcorner t_x\sigma\urcorner = \ulcorner t_x\sigma'\urcorner$. Since r is a possible candidate t_x is well-defined.

It is obvious that t_x can neither be a variable nor a constant. In fact, it is easy to see that t_x must be of the form $Exp(t_1, t_2^{z_2} \cdot \ldots \cdot t_k^{z_k})$. By Definition 2.6, 3. and since $x \in \mathcal{V}(t_x)$, for all $j \in \{1, \ldots, k\}$ but one the t_j are ground terms. Let j_0 be

the index of the non-ground term, i.e., $x \in \mathcal{V}(t_{j_0})$. By minimality of t_x, we have $\ulcorner t_{j_0}\sigma \urcorner \neq \ulcorner t_{j_0}\sigma' \urcorner$. Recalling that $\ulcorner t_x\sigma \urcorner = \ulcorner t_x\sigma' \urcorner$, we consider the possible cases:

—$j_0 = 1$: Let us first assume that $\ulcorner t_{j_0}\sigma \urcorner = \mathit{Exp}(b, M)$ and $\ulcorner t_{j_0}\sigma' \urcorner = \mathit{Exp}(b', M')$. Since $\ulcorner t_x\sigma \urcorner = \ulcorner t_x\sigma' \urcorner$, we must have $b = b'$ and $\ulcorner M \cdot \Pi_{i=2}^n t_i^{z_i} \urcorner = \ulcorner M' \cdot \Pi_{i=2}^n t_i^{z_i} \urcorner$. This implies $M = M'$ and therefore $\ulcorner t_{j_0}\sigma \urcorner = \ulcorner t_{j_0}\sigma' \urcorner$, in contradiction to the assumption. If both $\ulcorner t_{j_0}\sigma \urcorner$ and $\ulcorner t_{j_0}\sigma' \urcorner$ are not of the form $\mathit{Exp}(\cdot, \cdot)$, then because of $\ulcorner t_x\sigma \urcorner = \ulcorner t_x\sigma' \urcorner$ it is again easy to see that $\ulcorner t_{j_0}\sigma \urcorner = \ulcorner t_{j_0}\sigma' \urcorner$. If $\ulcorner t_{j_0}\sigma \urcorner = \mathit{Exp}(b, M)$ but $\ulcorner t_{j_0}\sigma' \urcorner$ is not of the form $\mathit{Exp}(\cdot, \cdot)$, then $\ulcorner t_x\sigma \urcorner = \ulcorner t_x\sigma' \urcorner$ implies that $\ulcorner M \cdot \Pi_{i=2}^n t_i^{z_i} \urcorner = \ulcorner \Pi_{i=2}^n t_i^{z_i} \urcorner$, and thus, $M = 1$, in contradiction to the fact that $\ulcorner t_{j_0}\sigma \urcorner$ is normalized. Hence, the case $j_0 = 1$ cannot occur.

—$j_0 > 1$: First assume that $t_1 = \mathit{Exp}(b, M)$. The equality $\ulcorner t_x\sigma \urcorner = \ulcorner t_x\sigma' \urcorner$ now implies that

$$\ulcorner M \cdot \ulcorner t_{j_0}^{z_{j_0}}\sigma \urcorner \cdot \Pi_{j \in \{2,\ldots,k\} \setminus \{j_0\}} t_j^{z_j} \urcorner = \ulcorner M \cdot \ulcorner t_{j_0}^{z_{j_0}}\sigma' \urcorner \cdot \Pi_{j \in \{2,\ldots,k\} \setminus \{j_0\}} t_j^{z_j} \urcorner.$$

It follows that $\ulcorner t_{j_0}^{z_{j_0}}\sigma \urcorner = \ulcorner t_{j_0}^{z_{j_0}}\sigma' \urcorner$, in contradiction to the assumption. The case that $\ulcorner t_x\sigma \urcorner$ is not of the form $\mathit{Exp}(\cdot, \cdot)$ is even simpler. □

We now use this lemma to prove that any subterm of a substitution of a minimal attack is either part of the protocol specification or a subterm of a normalized message transmitted during the attack.

LEMMA 3.9. *Let (π, σ) be an attack on P with σ normalized, $x \in \mathcal{V}(R_i)$ for some $i \in \{1, \ldots, k\}$, and $s \in \mathcal{S}(\sigma(x))$ standard. Then, there exists $j \leq i$ such that $s \in \mathcal{S}(\ulcorner R_j\sigma \urcorner)$ or there exists $t \in \mathcal{S}(P)$ with $t \sqsubseteq_\sigma s$.*

PROOF. Assume that there does not exist $t \in \mathcal{S}(P)$ with $t \sqsubseteq_\sigma s$ and $s \notin \mathcal{S}(\ulcorner R_j\sigma \urcorner)$ for all $j \leq i$. It follows that s is not an atom. Let $V_j = \mathcal{V}(R_j) \cup \mathcal{V}(S_j)$ and let j be minimal such that $s \in \mathcal{S}(V_j\sigma)$. By Definition 2.6, 2. we have $s \in \mathcal{S}(\mathcal{V}(R_j)\sigma)$. Let $\sigma' = \ulcorner \sigma[s \leftarrow 1] \urcorner$.

By minimality of j, for all $l < j$ we have $R_l\sigma' = R_l\sigma$, and thus, $\ulcorner R_l\sigma' \urcorner = \ulcorner R_l\sigma \urcorner$. Since $s \notin \mathcal{S}(\ulcorner R_j\sigma \urcorner)$ we also have $\ulcorner R_j\sigma \urcorner[s \leftarrow 1] = \ulcorner R_j\sigma \urcorner$. Now, Lemma 3.5 implies $\ulcorner R_j\sigma' \urcorner = \ulcorner R_j\sigma \urcorner$. Note that σ and σ' differ on at least one variable in $\mathcal{V}(R_1, \ldots, R_j)$. This contradicts Lemma 3.8. □

3.2.3 *Properties of Derivations.* Here we show some useful properties of derivations which will allow us to easily replace terms in derivations. Let us start with a simple observation which easily follows from the definition of decomposition and composition rules.

LEMMA 3.10. *For every normalized finite set E of messages, message t, t-rule L with $E \rightarrow_L E, t$ and $\mathcal{S}(E, t) \neq \mathcal{S}(E)$, it follows that $\mathcal{S}(E, t) = \mathcal{S}(E) \cup \{t\}$ and L is a composition rule.*

The next lemma states that if a term t' is a subterm of a term t and t can be derived from a set E but t' is not a subterm of E, then t' can be derived from E and the last step of the derivation is a composition rule.

LEMMA 3.11. *Assume that $t' \in \mathcal{S}(t) \setminus \mathcal{S}(E)$ and $t \in \mathit{forge}(E)$, then $t' \in \mathit{forge}(E)$ and there exists a derivation from E with goal t' ending with a composition rule.*

PROOF. Let $D = E_0 \rightarrow_{L_1} E_1 \cdots \rightarrow_{L_n} E_n$ be a derivation of t from $E_0 = E$. Then, since t' is a subterm of E_n there exists $i > 0$ minimal such that $t' \in \mathcal{S}(E_i)$. By minimality of i we have $t' \in \mathcal{S}(E_i) \setminus \mathcal{S}(E_{i-1})$. Using Lemma 3.10, it follows that $D = E_0 \rightarrow_{L_1} E_1 \cdots \rightarrow_{L_i} E_i$ is a derivation with goal t'. □

The following lemma is used in the proof of Lemma 3.13. It allows the construction of special derivations where a given term is never decomposed. This will be critical to replace composed terms by atoms in some derivations.

LEMMA 3.12. *Let $t \in forge(E)$ and $\gamma \in forge(E)$ be given with a derivation D_γ from E ending with an application of a rule in $\mathcal{L}_c$. Then, there is a derivation D' from E with goal t satisfying $L_d(\gamma) \notin D'$.*

PROOF. See Appendix 9.1. □

We can now prove a lemma which will allow us to replace certain subterms occurring in a substitution of an attack by smaller terms. Note that from the assumption made in this lemma it follows that s can be derived from E such that the last rule is a composition rule. This allows us to replace s by a smaller term since when deriving t, decomposing s will not be necessary.

LEMMA 3.13. *Let E and F be two sets of normalized messages such that $1 \in E \cup F$. Let $t \in forge(E, F)$ and $s \in forge(E)$ such that s is non-atomic and $s \notin \mathcal{S}(E)$. Finally, let δ be the replacement $[s \leftarrow 1]$. Then, $\ulcorner t\delta \urcorner \in forge(\ulcorner E\delta, F\delta \urcorner)$.*

PROOF. By Lemma 3.11 there exists a well-formed derivation D_s from E with goal s such that the last step is a composition rule. By Lemma 3.12, there exists a derivation D_t from E, F with goal t such that $L_d(s) \notin D_t$. Assume that $D_t = E, F \rightarrow_{L_1} E, F, t_1 \rightarrow_{L_2} E, F, t_2 \cdots \rightarrow_{L_n} E, F, t_1 \ldots, t_n$. We show $\ulcorner t_i\delta \urcorner \in forge(\ulcorner E\delta, F\delta \urcorner)$ by induction on $i \leq n$ where t_0 is some term in E, F. Induction base: If $t_0 \in E \cup F$, then clearly $\ulcorner t_0\delta \urcorner \in \ulcorner E\delta \cup F\delta \urcorner$. Induction step: We distinguish several cases.

—If $L_i = L_c(\langle a, b \rangle)$, then either $t_i = s$, and thus, $t_i\delta = 1 \in forge(E\delta, F\delta)$, or $t_i\delta = \langle a\delta, b\delta \rangle \in forge(\ulcorner E\delta, F\delta \urcorner)$ since by induction we have $\{a\delta, b\delta\} \subseteq forge(\ulcorner E\delta, F\delta \urcorner)$. Analogously, the terms $\{a\}^s_b$ and $\{a\}^p_K$ are treated.

—If $L_i = L_{p1}(\langle t_i, a \rangle)$, then $s \neq \langle t_i, a \rangle$ since $L_i \notin L_d(s)$. Therefore, $\langle t_i, a \rangle \delta = \langle t_i\delta, a\delta \rangle$. By induction, $\langle t_i, a \rangle \delta \in forge(E\delta, F\delta)$, and thus, $t_i\delta \in forge(E\delta,\ F\delta)$. Analogously, the cases for L_{p2}, L_{sd}, and L_{ad} are shown.

—If $L_i \in L_o$, then we use Definition 2.11, 3. Let E' be the set of messages obtained in D_s before the last step is applied. Then, $E' \setminus s \rightarrow_{\mathcal{L}_c(s)} E', s$. Also, $E, F, t_1, \ldots, t_{i-1} \rightarrow_{L_i} E, F, t_1, \ldots, t_{i-1}, t_i$. In particular $E', E, F, t_1, \ldots, t_{i-1} \rightarrow_{L_i} E', E, F, t_1, \ldots, t_{i-1}, t_i$. By Definition 2.11, 3. we obtain:

$$\ulcorner t_i\delta \urcorner \in forge(\ulcorner E'\delta, E\delta, F\delta, t_1\delta, \ldots, t_{i-1}\delta \urcorner)$$

We know that $E'\delta = E'$, $E\delta = E$, and all terms in E' can be derived from E. Also, E and E' are normalized. By induction, $\ulcorner t_1\delta \urcorner, \ldots, \ulcorner t_{i-1}\delta \urcorner \in forge(\ulcorner E\delta, F\delta \urcorner)$. Thus, $forge(\ulcorner E'\delta, E\delta, F\delta, t_1\delta, \ldots, t_{i-1}\delta \urcorner) \subseteq forge(\ulcorner E\delta, F\delta \urcorner)$, and therefore, $\ulcorner t_i\delta \urcorner \in forge(\ulcorner E\delta, F\delta \urcorner)$

For $i = n$, this gives us $t\delta \in forge(E\delta, F\delta)$. □

3.2.4 *Properties of Minimal Attacks.* We are now ready to prove Proposition 3.14 which states that substitutions of minimal attacks can always be constructed by linking subterms that are initially occurring in the protocol P. This will be the key to bound the number of subterms in minimal attacks (Theorem 3.15).

PROPOSITION 3.14. *Let (π, σ) be a minimal attack on the protocol P. Then, for all $s \in \mathcal{S}(\mathcal{V}\sigma)$ there exists $t \in \mathcal{S}(P)$ such that $t \sqsubseteq_\sigma s$.*

PROOF. Let (π, σ) and s be as above. Assume (*): For every t, $t \sqsubseteq_\sigma s$ implies $t \notin \mathcal{S}(P)$. We will lead this to a contradiction. Since $\mathcal{A} \subseteq \mathcal{S}(P)$, we have $s \notin \mathcal{A}$. By Lemma 3.9 and (*), there exists j such that $s \in \mathcal{S}(\lceil R_j\sigma \rceil)$. Let N be minimal among the possible j. If $s \in \mathcal{S}(\lceil S_i\sigma \rceil)$ for some i, (*) together with Lemma 3.7 imply that there exists $x \in \mathcal{V}(S_i)$ with $s \in \mathcal{S}(x\sigma)$. Then, by Definition 2.6, (2) there exists $R_{i'}, i' \leq i$ such that $x \in \mathcal{V}(R_{i'})$. Thus, Lemma 3.9 and (*) imply that there exists $j \leq i$ with $s \in \mathcal{S}(\lceil R_j\sigma \rceil)$. Note also that $s \notin \mathcal{S}(S_0)$ since otherwise $s \in \mathcal{S}(P)$. Now, the minimality of N yields $i \geq N$. Let $E_j = \lceil S_0\sigma, \ldots, S_{j-1}\sigma \rceil$.

Summarizing, we have: s is non-atomic, not a subterm of E_N but a subterm of $\lceil R_N\sigma \rceil$. Thus, by Lemma 3.11, $s \in \mathit{forge}(E_N)$.

Let δ be the replacement $[s \leftarrow 1]$. Since (π, σ) is an attack, we have for all $1 \leq j \leq k+1$ and $R_{k+1} = \mathsf{secret}$:

$$\lceil R_j\sigma \rceil \in \mathit{forge}(E_j)$$

We distinguish two cases:

—Assume $j < N$. Then, by minimality of N, s is neither a subterm of $\lceil R_j\sigma \rceil$ nor a subterm of E_j. Hence, with $\lceil R_j\sigma \rceil \in \mathit{forge}(E_j)$ it follows $\lceil\lceil R_j\sigma \rceil\delta \rceil \in \mathit{forge}(\lceil E_j\delta \rceil)$.

—Assume $j \geq N$. With $t = \lceil R_j\sigma \rceil$, $E = E_N$, and $F = E_j$, Lemma 3.13 implies $\lceil\lceil R_j\sigma \rceil\delta \rceil \in \mathit{forge}(\lceil E_j\delta \rceil)$.

Thus, $\lceil\lceil R_j\sigma \rceil\delta \rceil \in \mathit{forge}(\lceil E_j\delta \rceil)$ in both cases. Now, (*) and Lemma 3.5 imply for all j:

$$\lceil R_j\sigma' \rceil \in \mathit{forge}(\lceil S_0\sigma', \ldots, S_{j-1}\sigma' \rceil)$$

where $\sigma' = \lceil \sigma\delta \rceil$. Hence, (π, σ') is an attack. (Note that the conditions for applying Lemma 3.5 are satisfied.) But since σ' is obtained from σ by replacing s by a strictly smaller message, namely 1, we obtain $|\sigma'| < |\sigma|$, a contradiction to the assumption that (π, σ) is a minimal attack. □

3.3 Bounding the Number of Subterms of Minimal Attacks

Using Proposition 3.14, we obtain:

THEOREM 3.15. *For every minimal attack (π, σ) of a protocol P it follows that $\mathcal{S}(\lceil \mathcal{S}(P)\sigma \rceil) = \lceil \mathcal{S}(P)\sigma \rceil$*

PROOF. The inclusion $\lceil \mathcal{S}(P)\sigma \rceil \subseteq \mathcal{S}(\lceil \mathcal{S}(P)\sigma \rceil)$ is trivial. The converse inclusion is a direct consequence of Proposition 3.14, which implies:

$$\mathcal{S}(\mathcal{V}\sigma) \subseteq \lceil \mathcal{S}(P)\sigma \rceil.$$

By Lemma 2.4, we know that $\mathcal{S}(\lceil \mathcal{S}(P)\sigma \rceil) \subseteq \lceil \mathcal{S}(\mathcal{S}(P)\sigma) \rceil$ and $\mathcal{S}(\mathcal{S}(P)\sigma) \subseteq \mathcal{S}(P)\sigma \cup \mathcal{S}(\mathcal{V}\sigma)$. Thus, $\mathcal{S}(\lceil \mathcal{S}(P)\sigma \rceil) \subseteq \lceil \mathcal{S}(P)\sigma \rceil \cup \mathcal{S}(\mathcal{V}\sigma)$, and with the above:

$$\mathcal{S}(\lceil \mathcal{S}(P)\sigma \rceil) \subseteq \lceil \mathcal{S}(P)\sigma \rceil. \quad \square$$

From this theorem, we immediately obtain:

COROLLARY 3.16. *For every minimal attack (π, σ) of a protocol P and every $E \subseteq \mathcal{S}(P)$ it follows that $|\ulcorner E\sigma \urcorner| \leq |P|$. In particular, $|\mathcal{V}(P)\sigma| \leq |P|$.*

PROOF. First, observe that the cardinality of the set $\ulcorner \mathcal{S}(P)\sigma \urcorner$ is bounded by $|P|$. Second, observe that $\ulcorner E\sigma \urcorner \subseteq \ulcorner \mathcal{S}(P)\sigma \urcorner$. From this, $|\ulcorner E\sigma \urcorner| \leq |P|$ follows immediately. We obtain $|\mathcal{V}(P)\sigma| \leq |P|$ if we set $E = \mathcal{V}(P)$. Note that σ is normalized, and thus, $\sigma(x) = \ulcorner \sigma(x) \urcorner$ for every variable x. □

4. EXTENDING THE DOLEV-YAO INTRUDER BY DIFFIE-HELLMAN EXPONENTIATION

We now extend the Dolev-Yao intruder given by the intruder rules $L_c \cup L_d$ (see Subsection 2.2) by a set L_o of rules, the DH rules, which allow the intruder to perform Diffie-Hellman exponentiation. This extended intruder is called the DH intruder. We want to show that for the DH intruder the derivation problem can be decided in deterministic polynomial time and the insecurity problem in non-deterministic polynomial time. To this end, we show that the preconditions of Theorem 3.2 and Theorem 3.1 are satisfied, respectively. Recall that Theorem 3.2 requires that i) L_o is a set of oracle rules and ii) OracleRule can be decided in polynomial time. In addition, Theorem 3.1 requires that iii) L_o allows polynomial product exponent attacks.

In Section 4.1 and 4.2, we prove i) and in Section 4.3 we prove ii). Using Theorem 3.2, we then conclude that for the DH intruder the derivation problem can be decided in deterministic polynomial time (Corollary 4.9).

In Section 5, we prove iii), which with Theorem 3.1 yields that for the DH intruder the insecurity problem can be decided in non-deterministic polynomial time (Theorem 5.23).

First, we define the DH rules L_o.

Definition 4.1. We define $L_o = L_{oc} \cup L_{od}$ to be the set of *DH rules* of the form

$$t, t_1, \ldots, t_n \to \ulcorner Exp(t, t_1^{z_1} \cdots t_n^{z_n}) \urcorner =: u$$

with $n \geq 1$, $z_i \in \mathbb{Z} \setminus \{0\}$, $1 \leq i \leq n$, $t, t_1, \ldots, t_n$ normalized standard messages. If u is of the form $Exp(\cdot, \cdot)$, then the above rule belongs to $L_{oc}(u)$ (the set of *composition DH rules*), and to $L_{od}(u)$ (the set of *decomposition DH rules*) otherwise. We call the intruder using the rules L_o as oracle rules the *DH intruder*. We call t in the above DH rule the *head* of this rule and we refer to $z_1, \ldots, z_n$ as the *product exponents* of this rule. We require w.l.o.g. that the head t of a decomposition DH rule is of the form $Exp(\cdot, \cdot)$ since otherwise $t = u$. Also, w.l.o.g. we may assume that $t_i \neq t_j$ for every $i \neq j$, and that $z_i \neq 0$ for every i.

Using that the messages on the right-hand side of a DH rule are normalized, one easily observes the following.

LEMMA 4.2. *The rules in L_{oc} and L_{od} are composition and decomposition guess rules, respectively.*

4.1 Diffie-Hellman Rules allow Well-formed Derivations

We show that L_o allows well-formed derivations (Lemma 4.5). In other words, we show that L_o satisfies the first property required of oracle rules (see Definition 2.11). The proof uses two lemmas, which are stated next.

The first lemma allows us to restrict our attention to certain kinds of derivations where the L_o rules are only used on messages which were created by Dolev-Yao rules or were present in the initial set of messages.

LEMMA 4.3. *Let E be a finite set of normalized standard messages and t be a standard message such that t can be derived from t (w.r.t. $\mathcal{L}$). Let D be a derivation from E with goal t. Then, there exists a derivation D' from E with goal t such that*

(1) D' is of the same length as D, and

(2) for every DH rule $L \in D' \cap L_o$ with head t' we have that $t' \in E$ or there exists a t'-rule $L' \in D' \cap (L_d \cup L_c)$. Moreover, if L is a decomposition DH rule, then $t' \in E$ or there exists a t'-rule $L' \in D' \cap L_d$.

PROOF. See Appendix 9.2. □

The next lemma gives us a criterion to determine whether a derivation is well-formed.

LEMMA 4.4. *Let $D = E_0 \rightarrow_{L_1} \dots E_{n-1} \rightarrow_{L_n} E_n$ be a derivation with goal g.*

(1) Assume that for every j with $E_{j-1} \rightarrow_{L_j} E_{j-1}, t$ the jth step in D and $L_j \in \mathcal{L}_d(t)$, there exists $t' \in E_{j-1}$ such that t is a subterm of t' and either $t' \in E_0$ or there exists i with $i < j$ and $L_i \in \mathcal{L}_d(t')$. Then, if $L \in D \cap \mathcal{L}_d(t)$ for some L and t, then $t \in \mathcal{S}(E_0)$.

(2) Assume that for every $i < n$ and t with $L_i \in \mathcal{L}_c(t)$, there exists j with $i < j$ such that L_j is a t'-rule and $t \in \mathcal{S}(\{t'\} \cup E_0)$. Then, if $L \in D \cap \mathcal{L}_c(t)$ for some L and t, then $t \in \mathcal{S}(E_0, g)$.

Given both the assumptions in 1. and 2., it follows that D is a well-formed derivation with goal g.

PROOF. See Appendix 9.2. □

We can now prove that the L_o rules allow well-formed derivations:

LEMMA 4.5. *For every finite normalized set E of standard messages and normalized standard message g, $g \in forge(E)$ implies that there exists a well-formed derivation from E with goal g.*

PROOF. Let $E_0 = E$ and $D = E_0 \rightarrow_{L_1} \dots \rightarrow_{L_n} E_n$ be a derivation of goal g of minimal length. We may assume that D satisfies the properties stated in Lemma 4.3, 2.

We prove that D satisfies the assumptions Lemma 4.4, 1. and 2.

(1) If $L_j \in L_d(s) \cap \mathcal{L}_d(t)$, and thus, $t \in \mathcal{S}(s)$, then $L_i \notin L_{oc}(s)$, for all $i < j$, since rules in L_{oc} do not create standard terms, and $L_i \notin L_c(s)$, for all $i < j$, by the definition of derivation (since otherwise t would be in the left-hand side of L_i). Therefore, either $s \in E_0$ or there exists $i < j$ with $L_i \in \mathcal{L}_d(s)$.

If $L_j \in L_{od}(t)$ and t' is the head of L_j, by definition of decomposition DH rules it is easy to see that $t \in \mathcal{S}(t')$. By Lemma 4.3, 2. it follows that $t' \in E_0$ or there exists $L' \in D \cap L_d(t')$.

By Lemma 4.4, 1. it follows that if $L \in D \cap \mathcal{L}_d(t)$ for some L and t, then $t \in \mathcal{S}(E_0)$.

(2) If $L_i \in \mathcal{L}_c(t)$ and $i < n$, then by minimality of D, there exists $j > i$ such that t belongs to the left-hand side of L_j. If $L_j \in L_d$, then as in 1. we can conclude that $t \in \mathcal{S}(E_0)$. If $L_j \in L_c(t')$, then $t \in \mathcal{S}(t')$. If $L_j \in L_o(t')$, then first assume that t is the head of L_j. Lemma 4.3, 2. implies that there exists a t-rule $L' \in D \cap (L_d \cup L_c)$. Since $L_i \in \mathcal{L}_c(t)$ and because of the minimality of D, t can only be generated by one rule, we have $L_i = L' \in L_c$. Thus, $t \neq \mathit{Exp}(\cdot,\cdot)$. But then, by definition of DH rules, $t \in \mathcal{S}(t')$. Now, assume that t is not the head of L_j. If $t \notin \mathcal{S}(t')$, we have that $t \in \mathcal{S}(t'')$ where t'' is the head of L_j and t'' is of the form $\mathit{Exp}(\cdot,\cdot)$. (Otherwise, t can not disappear from t'.) By Lemma 4.3, 2. it follows that $t'' \in E_0$ or there exists a t''-rule $L' \in D \cap (L_d \cup L_c)$. Since t'' is of the form $\mathit{Exp}(\cdot,\cdot)$ we know that $L' \in D \cap L_d$. Now, 1. implies that $t'' \in \mathcal{S}(E_0)$, and thus, $t \in \mathcal{S}(E_0)$. □

4.2 Diffie-Hellman Rules are Oracle Rules

We now prove the remaining properties required of oracle rules, and thus, show that L_o forms a set of oracle rules (Proposition 4.7). First, we need a lemma similar to Lemma 3.6.

LEMMA 4.6. *Let $z_1, \ldots, z_n \in \mathbb{Z} \setminus \{0\}$, and $s, s_1, \ldots, s_n, u$ be normalized standard terms such that $s_i \neq s_j$ for every $i \neq j$, $s_i \neq 1$ and $s_i \neq u$ for every i, $s \neq u$, $u = \ulcorner \mathit{Exp}(s, s_1^{z_1} \cdots s_n^{z_n}) \urcorner$, and $u = \mathit{Exp}(\cdot,\cdot)$. Let δ be the replacement $[u \to 1]$. Then, $u = \ulcorner \mathit{Exp}(\ulcorner s\delta \urcorner, \ulcorner s_1\delta \urcorner^{z_1} \cdots \ulcorner s_n\delta \urcorner^{z_n}) \urcorner$.*

PROOF. See Appendix 9.2. □

We are now prepared to prove:

PROPOSITION 4.7. *The set L_o of DH rules is a set of oracle rules.*

PROOF. We check each condition 1., 2., and 3. in Definition 2.11:

(1) This is an immediate consequence of Lemma 4.5.

(2) This follows from the observation that no term created with L_{oc} can be decomposed with L_d.

(3) Let u be a normalized standard message, F be a set of standard messages with $1 \in F$, and t be a standard message such that $F \setminus u \to_{\mathcal{L}_c(u)} F$ and $F \to_{L_o(t)} F, t$. Let $\delta := [u \leftarrow 1]$. If $u = t$, then $t\delta = 1 \in \mathit{forge}(F\delta)$, and we are done. Now, assume that $u \neq t$. Since $F \to_{L_o(t)} F, t$, there exist $t', t_1, \ldots, t_n \in F$ and $z_1, \ldots, z_n \in \mathbb{Z} \setminus \{0\}$ such that $t_i \neq t_j$ for every $i \neq j$ and $t = \ulcorner \mathit{Exp}(t', t_1^{z_1} \cdots t_n^{z_n}) \urcorner$. If $t' \neq \mathit{Exp}(\cdot,\cdot)$ or $u \neq t'$, then by Lemma 3.6 we obtain that

$$\ulcorner t\delta \urcorner = \ulcorner \mathit{Exp}(\ulcorner t'\delta \urcorner, \ulcorner t_1\delta \urcorner^{z_1} \cdots \ulcorner t_n\delta \urcorner^{z_n}) \urcorner.$$

Thus, $\ulcorner t\delta \urcorner \in \mathit{forge}(\ulcorner F\delta \urcorner)$. Now, assume that $u = t' = \mathit{Exp}(v, M)$. Then,

$$\begin{aligned}
\ulcorner t\delta \urcorner &= \ulcorner\urcorner \mathit{Exp}(v, M{\cdot}t_1^{z_1} \cdots t_n^{z_n})\urcorner\delta\urcorner \\
&= \ulcorner \mathit{Exp}(v, M{\cdot}t_1^{z_1} \cdots t_n^{z_n})\delta\urcorner && (*) \\
&= \ulcorner \mathit{Exp}(v\delta, M\delta{\cdot}(t_1\delta)^{z_1} \cdots (t_n\delta)^{z_n})\urcorner && (**) \\
&= \ulcorner \mathit{Exp}(v, M{\cdot}(t_1\delta)^{z_1} \cdots (t_n\delta)^{z_n})\urcorner && (***) \\
&= \ulcorner \mathit{Exp}(v, M{\cdot}\ulcorner t_1\delta\urcorner^{z_1} \cdots \ulcorner t_n\delta\urcorner^{z_n})\urcorner \\
&= \ulcorner \mathit{Exp}(u, \ulcorner t_1\delta\urcorner^{z_1} \cdots \ulcorner t_n\delta\urcorner^{z_n})\urcorner
\end{aligned}$$

where in (*) we apply Lemma 3.4, 2. using that $v \neq u$ and $u \neq t$. In (**) we again use that $u \neq t$. We obtain (***) since $u \notin \mathcal{S}(v, M)$.
Now, to show that $\ulcorner t\delta \urcorner \in \mathit{forge}(\ulcorner F\delta \urcorner)$ it suffices to show that $u \in \mathit{forge}(\ulcorner F\delta \urcorner)$. We know $F \setminus u \rightarrow_{\mathcal{L}_c(u)} F$, and since $u = \mathit{Exp}(\cdot, \cdot)$, we have $F \setminus u \rightarrow_{L_{oc}(u)} F$. Hence, there exist normalized terms $s, s_1, \ldots, s_n \in F \setminus u$ and $z'_1, \ldots, z'_{n'} \in \mathbb{Z} \setminus \{0\}$ such that s and the s_i meet the conditions stated in Lemma 4.6 and $u = \ulcorner \mathit{Exp}(s, s_1^{z_1} \cdots s_n^{z_n})\urcorner$. Then, by Lemma 4.6, $u = \ulcorner \mathit{Exp}(\ulcorner s\delta\urcorner, \ulcorner s_1\delta\urcorner^{z_1} \cdots \ulcorner s_n\delta\urcorner^{z_n})\urcorner$. Thus, $u \in \mathit{forge}(\ulcorner F\delta \urcorner)$. □

4.3 Deciding DH Rules

The following proposition states that it is decidable in polynomial time whether a given message can be derived from a finite set of messages by applying an oracle rule once.

PROPOSITION 4.8. *For the DH intruder, the problem* ORACLERULE *is decidable in deterministic polynomial time.*

PROOF. We need to show that there is a deterministic polynomial time algorithm that given E and t decides whether there exists $t', t_1, \ldots, t_n \in E$ and $z_1, \ldots, z_n \in \mathbb{Z}$ such that $t = \ulcorner \mathit{Exp}(t', t_1^{z_1} \cdots t_n^{z_n})\urcorner$. It is easy to see that $E \rightarrow_{L_o} E, t$ iff

(1) $t \neq \mathit{Exp}(\cdot, \cdot)$ and
  (a) $t \in E$, or
  (b) there exists M with $\mathit{Exp}(t, M) \in E$ and $\mathcal{F}(M) \subseteq E$, or
(2) $t = \mathit{Exp}(v, M)$ and
  (a) $v \in E$ and $\mathcal{F}(M) \subseteq E$, or
  (b) there exists M' such that $\mathit{Exp}(v, M') \in E$ and $E' := \{t' \mid$ the product exponents in M and M' for t' differ$\} \subseteq E$.

From this characterization of $E \rightarrow_{L_o} E, t$ it is straightforward to derive a polynomial time algorithm for deciding $E \rightarrow_{L_o} E, t$. □

As an immediate consequence of the above proposition, Proposition 4.7, and Theorem 3.2, we obtain the following corollary:

COROLLARY 4.9. *For the DH intruder,* DERIVE *can be decided in deterministic polynomial time.*

5. THE DH RULES ALLOW POLYNOMIAL PRODUCT EXPONENT ATTACKS

In this section, we show that INSECURE is NP-complete for the DH intruder (Theorem 5.23). By Theorem 3.1, Proposition 4.7, and Proposition 4.8, it remains to

show that DH rules allow polynomial product exponent attacks. To this end, we will associate with a minimal attack (π, σ) a substitution σ^Z and a linear equation system such that i) σ^Z coincides with σ except that the product exponents in σ are replaced by new (integer) variables and ii) (π, σ') is an attack for every σ' obtained from σ^Z by substituting the variables in σ^Z according to a solution of the equation system. Since the size of the linear equation system can be bounded polynomially in the size of the protocol, and thus, the size of the solutions of this equation system can be bounded polynomially (see [Bockmayr and Weispfenning 2001]), we obtain an attack with polynomially bounded product exponents (Proposition 5.22).

In the following subsection, we define messages that may have linear expressions as product exponents. Before going into more detail, in Section 5.2, we provide some more intuition behind the proof of Proposition 5.22. A detailed proof is then given in Section 5.3 to 5.7.

5.1 Open Messages and Equation Systems

In this section, we define open messages and products, evaluation mappings, and equation systems as well as various measures on the size of these objects.

Definition 5.1. 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, 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 to represent e where integers are encoded in binary. We say that e and e' are *equal* if they are equal modulo associativity and commutativity of addition (modulo AC_+, for short). In particular, for a set of linear expressions S and a linear expression e, we say that e belongs to S if the equivalence class modulo AC_+ of e is one of the equivalences classes induced by S. In the same way, the subset relationship between sets of linear expressions is defined.

For an open message or an open product t let $\mathcal{L}_{exp}(t)$ denote the set of linear expressions occurring in t. The set $\mathcal{S}(t)$ of subterms of t and $|t|$, i.e., the number of subterms of t, are defined as usual. Also, recall that set $\mathcal{S}_{ext}(t)$ of *extended subterms* of t is defined as $\mathcal{S}(t) \cup \{M \mid Exp(u, M) \in \mathcal{S}(t)\}$. We define $|t| = \mathsf{Card}(\mathcal{S}(t))$ and $|t|_{ext} = \mathsf{Card}(\mathcal{S}_{ext}(t))$. Also, we set $|t|_{exp} = 0$ if t is not a product and $|t|_{exp} = |e_1| + \ldots + |e_n|$ if $t = t_1^{e_1} \ldots t_n^{e_n}$. As usual, if E is a finite set of open messages or products, $|E|_{exp} = \Sigma_{s \in E} |s|_{exp}$. With this we define $||t||_{exp} = |\mathcal{S}(t)|_{exp}$. Finally, $||t|| = |t| + ||t||_{exp}$, and $||t||_{ext} = |t|_{ext} + ||t||_{exp}$. We note :

LEMMA 5.2. *For every open message or product t we have that $|t|_{ext} \leq 2 \cdot |t|$, and thus, $||t||_{ext} \leq 2 \cdot ||t||$.*

Note that the definitions of $|\cdot|$, $||\cdot||_{exp}$, and $||\cdot||$ for open messages and products correspond to those for messages.

The above definitions and measures for open messages and products extend in the obvious way to sets of open messages, open products, etc.

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 usual. The evaluation mapping β extends in the obvious way to open messages, open products, sets of open messages, etc.

Throughout this section, β will always denote an evaluation mapping from Z into $\mathbb{Z}$.

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. 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 $e = e' \in \mathcal{E}$. Let $\mathcal{L}_{exp}(\mathcal{E}) = \{e \mid e = e' \in \mathcal{E} \text{ or } e' = e \in \mathcal{E}\}$ denote the set of linear expressions occurring in $\mathcal{E}$. Let $R_{\mathcal{E}} = \{(e, e') \mid e = e' \in \mathcal{E}\} \subseteq \mathcal{L}_{exp}(\mathcal{E}) \times \mathcal{L}_{exp}(\mathcal{E})$ and let $R^*_{\mathcal{E}}$ denote the reflexive and transitive closure of $R_{\mathcal{E}}$. We write $\mathcal{E} = \mathcal{E}'$ if $R^*_{\mathcal{E}} = R^*_{\mathcal{E}'}$. We write $\mathcal{E} \subseteq \mathcal{E}'$ if $R^*_{\mathcal{E}} \subseteq R^*_{\mathcal{E}'}$. Thus, we consider linear equations modulo reflexivity and transitivity of equality. Recall also that linear expressions are considered modulo AC_+.

5.2 Overview of the Proof of Proposition 5.22

To provide an overview of the proof of Proposition 5.22, we give an informal top-down view of the proof. Recall that this proposition allows us to bound the product exponent size of attacks.

The key for the proof is Lemma 5.21, which states: Let $t, t_1, \ldots, t_n$ be open messages and β be an evaluation mapping such that $\ulcorner \beta(t) \urcorner \in \mathit{forge}(\ulcorner \beta(t_1) \urcorner, \ldots, \ulcorner \beta(t_n) \urcorner)$. Then, there exists an extension of β (also called 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 $||t_1, \ldots, t_n, t||_{ext}$.

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 (or β-terms), such that $\beta(t'_i) = \ulcorner \beta(t_i) \urcorner$. In other words, t'_i is the symbolic representation of the normal form of $\beta(t_i)$. Now, we can simulate 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.

More precisely, when turning t_i into β-normal forms, we also associate an equation system with this normal form, i.e., we define what we call a β-tuple $(t'_i, \mathcal{E}_i)$ where t'_i is the β-normal form of t_i and $\mathcal{E}_i$ is an equation system such that β is a solution of $\mathcal{E}_i$ and for every solution β' of $\mathcal{E}_i$ we have that $\ulcorner \beta'(t_i) \urcorner = \ulcorner \beta'(t'_i) \urcorner$. In other words, t'_i is not only the symbolic representation of the normal form of $\beta(t_i)$ but also the symbolic representation of the normal form of t_i for other evaluation mappings β', although in this case $\beta'(t'_i)$ needs to be normalized to coincide with $\ulcorner \beta'(t_i) \urcorner$. The equation system $\mathcal{E}$ is obtained as a union of the equation systems of the β-tuples for $t_1, \ldots, t_n, t$ and equations obtained in the course of simulating D.

Clearly, to prove the lemma, it is necessary to bound the size of β-tuples, i.e., of β-normal forms (β-terms) and the equations associated, as well as the size of equations obtained when simulating D (see the subsequent sections for details).

Using the above lemma, it is now rather easy to prove Proposition 5.22, which states that for every minimal attack (π, σ) there exists an attack (π, σ') of the same structure, i.e., σ and σ' coincide up to product exponents, such that the size of (all) the product exponents of σ' can be polynomially bounded in the size of the protocols.

The idea of the proof of Proposition 5.22 is simply to associate to σ a symbolic version σ^Z where all product exponents of σ are replaced by (new) integer variables. Then, we apply the above lemma to the case where $t = R_i\sigma^Z$ and $t_j = S_j\sigma^Z$ for every $j \in \{0, \ldots, i-1\}$. For every i, the lemma yields an equation system $\mathcal{E}'_i$ and the solutions β' of the union of these systems yield new attacks $(\pi, \beta'(\sigma^Z))$ on the protocol. Using that the (union of the) equation systems $\mathcal{E}'_i$ are "small" and the fact that linear equation systems have "small" solutions β' [Bockmayr and Weispfenning 2001], we obtain an attack (π, σ') with $\sigma' = \beta'(\sigma^Z)$ with "small" product exponents.

In the following section, we define β-tuples. We then show that β-tuples always exist (Section 5.4). In Section 5.5 and 5.6, we bound the size of β-terms and their associated equation systems, respectively, and thus, the size of β-tuples as a whole. Then, in Section 5.7 we prove the mentioned Lemma 5.21 and Proposition 5.22, and from this derive that INSECURE is NP-complete for the DH intruder.

5.3 β-equivalence, β-tuples, and $\approx_\beta$-equation Systems

Definition 5.3. Given β and open messages or open products t and t', we say that t and t' are *β-equal* ($t =_\beta t'$) iff $\beta(t) = \beta(t')$.[1] We call t and t' *β-equivalent* ($t \approx_\beta t'$) iff $\ulcorner\beta(t)\urcorner = \ulcorner\beta(t')\urcorner$.

Definition 5.4. Given β and open messages or open products t and t' such that $t =_\beta t'$, we say that $\mathcal{E}$ is a $=_\beta$-equation system for t and t' iff

(1) $\beta \models \mathcal{E}$ and

(2) $t =_{\beta'} t'$ for all $\beta' \models \mathcal{E}$.

We now show that "small" $=_\beta$-equation systems exist. Recall that, for instance, when we write $||t, t'||_{ext}$ we mean $||\{t, t'\}||_{ext}$.

LEMMA 5.5. *Given β and open messages or open products t and t' such that $t =_\beta t'$. Then there exists a $=_\beta$-equation system $\mathcal{E}^{=_\beta}_{t,t'}$ of size $\leq 2||t, t'||^3_{ext}$ for t and t'.*

PROOF. We define $R \subseteq \mathcal{S}_{ext}(t, t') \times \mathcal{S}_{ext}(t, t')$ such that $s =_\beta s'$ for all $(s, s') \in R$. More precisely, R is the smallest binary relation over $\mathcal{S}_{ext}(t, t')$ such that

—$(t, t') \in R$,

—If $(s, s') \in R$, and thus, by construction $s =_\beta s'$, and $s = \langle t_1, t_2\rangle$ it follows that $s' = \langle t'_1, t'_2\rangle$ for some open messages t'_1 and t'_2. Then, $\langle t_1, t'_1\rangle \in R$ and $\langle t_2, t'_2\rangle \in R$. For encryption we have analogous conditions on R.

[1] Recall that equality means equality modulo associativity and commutativity of multiplication in products, e.g., $a^2 \cdot b^3 \cdot c^{-2} = c^{-2} \cdot a^2 \cdot b^3$.

—If $(s, s') \in R$, and s is the product $t_1^{e_1} \cdots t_n^{e_n}$, $n \geq 1$, we have that s' is a product of the form $t'^{e'_1}_1 \cdots t'^{e'_n}_n$, $n \geq 1$. If $t_i =_\beta t'_j$ for some i and j, then $(t_i, t'_j) \in R$. Note that since $s =_\beta s'$, for every t_i there exists at least one $=_\beta$-equal term t'_j.

—If $t = Exp(u, M)$ for some open message u and an open product M, we have that $t' = Exp(u', M')$ for some open message u' and an open message M'. Then, $(u, u') \in R$ and $(M, M') \in R$.

For every $(s, s') \in R$, we define the equation system $\mathcal{E}_{(s,s')}$ as follows: If s (and thus, s') is not a product, then $\mathcal{E}_{(s,s')}$ is the empty set. Otherwise, s is of the form $t_1^{e_1} \cdots t_n^{e_n}$, $n \geq 1$, and s' is of the form $t'^{e'_1}_1 \cdots t'^{e'_n}_n$. We define $\mathcal{E}_{(s,s')} = \{e_i = e'_j \mid (t_i, t'_j) \in R\}$.

By structural induction it is easy to see that $\mathcal{E}_R = \bigcup_{(s,s') \in R} \mathcal{E}_{(s,s')}$ is a $=_\beta$-equation system for t and t'. Obviously, the size of $\mathcal{E}_R$ is $\leq 2||t, t'||^3_{ext}$. □

In what follows, we refer to $\mathcal{E}^{=_\beta}_{t,t'}$, as defined in the proof, as the *$=_\beta$-equation system induced by t and t'*.

Remark 5.6. The equation system $\mathcal{E}^{=_\beta}_{t,t'}$ as constructed in Lemma 5.5 is uniquely determined.

We will also need to associate with $\approx_\beta$-equivalent terms an equation system, which we call a $\approx_\beta$-equation system.

Definition 5.7. Given β and open messages or open products t and t' such that $t \approx_\beta t'$, we say that $\mathcal{E}$ is a $\approx_\beta$-equation system for t and t' iff

(1) $\beta \models \mathcal{E}$ and
(2) $t \approx_{\beta'} t'$ for all $\beta' \models \mathcal{E}$.

To construct such an equation system given t and t', we introduce the notion of a β-tuple.

Definition 5.8. Given β and an open message or open product t we say that $(t', \mathcal{E})$ where t' is an open message or an open product and $\mathcal{E}$ is an equation system is a *β-tuple* for t iff

(1) $\beta(t') = \ulcorner \beta(t) \urcorner$,
(2) $\beta \models \mathcal{E}$, and
(3) $t \approx_{\beta'} t'$ for every $\beta' \models \mathcal{E}$.

We call t' a *β-term* (or the *β-normal form*) of t and $\mathcal{E}$ a *β-equation system* for t.

The following lemma shows how a $\approx_\beta$-equation system can be obtained using β-tuples.

Lemma 5.9. *Let t and t' be open messages or open products such that $t \approx_\beta t'$. Assume that there exists a β-tuple $(s, \mathcal{E})$ for t and a β-tuple $(s', \mathcal{E}')$ for t'. Let $\mathcal{E}^{=_\beta}_{s,s'}$ be a $=_\beta$-equation system for s and s' (such a system always exists). Then,*

$$\mathcal{E}^{\approx_\beta}_{t,t'} = \mathcal{E} \cup \mathcal{E}' \cup \mathcal{E}^{=_\beta}_{s,s'}$$

is a $\approx_\beta$-equation system for t and t'.

PROOF. We first show that there always exists a $=_\beta$-equation for s and s'. We have that $\beta(s) = \ulcorner\beta(t)\urcorner = \ulcorner\beta(t')\urcorner = \beta(s')$. Thus, $s =_\beta s'$ and by Lemma 5.5 there exists a $=_\beta$-equation system, which we call $\mathcal{E}^{=_\beta}_{s,s'}$.

We need to show that $\mathcal{E}^{\approx_\beta}_{t,t'}$ is a $\approx_\beta$-equation system for t and t'. Obviously, $\beta \models \mathcal{E}^{\approx_\beta}_{t,t'}$. Let $\beta' \models \mathcal{E}^{\approx_\beta}_{t,t'}$. We need to show that $\ulcorner\beta'(t)\urcorner = \ulcorner\beta'(t')\urcorner$. Since $\beta' \models \mathcal{E}^{=_\beta}_{s,s'}$ we know $\beta(s) = \beta(s')$. From $\beta' \models \mathcal{E}$ ($\beta' \models \mathcal{E}'$) we conclude $\ulcorner\beta'(s)\urcorner = \ulcorner\beta'(t)\urcorner$ ($\ulcorner\beta'(s')\urcorner = \ulcorner\beta'(t')\urcorner$). Thus, $\ulcorner\beta'(t)\urcorner = \ulcorner\beta'(s)\urcorner = \ulcorner\beta'(s')\urcorner = \ulcorner\beta'(t')\urcorner$. □

5.4 Existence of β-tuples

We show that β-tuples exist for any open message and product.

LEMMA 5.10. *Let t be an open message or an open product and β be an evaluation mapping. Then, there exists a β-tuple for t.*

PROOF. We construct a β-tuple $(t^\beta, \mathcal{E}^\beta_t)$ of t inductively.

— If $t \in \mathcal{A}$, then $t^\beta = t$ and $\mathcal{E}^\beta_t = \emptyset$. Obviously, $(t^\beta, \mathcal{E}^\beta_t)$ is a β-tuple for t.

— If $t = \langle t_1, t_2\rangle$, let $(t^\beta_1, \mathcal{E}^\beta_{t_1})$ and $(t^\beta_2, \mathcal{E}^\beta_{t_2})$ be β-tuples for t_1 and t_2. We define $t^\beta = \left\langle t^\beta_1, t^\beta_2 \right\rangle$ and $\mathcal{E}^\beta_t = \mathcal{E}^\beta_{t_1} \cup \mathcal{E}^\beta_{t_2}$. Analogously, β-tuples are constructed in case of encryption. By induction, it is easy to see that $(t^\beta, \mathcal{E}^\beta_t)$ is a β-tuple for t.

— If $t = t_1^{e_1} \cdots t_n^{e_n}$, then let $(t^\beta_i, \mathcal{E}^\beta_{t_i})$ be the β-tuple for t_i for every i. Assume that $C_1, \ldots, C_l \subseteq \{t_1, \ldots, t_n\}$ are the equivalence classes over $t_1, \ldots, t_n$ modulo $\approx_\beta$. Define $e_{C_j} = \Sigma_{t_i \in C_j} e_i$ and let $s_{C_j} \in C_j$ be some representative of C_j. W.l.o.g. assume that for $s \in C_1$ we have $s \approx_\beta 1$. (The set C_1 may be empty.) By induction we have that $s^\beta = 1$ for every $s \in C_1$ since $\beta(s^\beta) = \ulcorner\beta(s)\urcorner = 1$. Define $J = \{j \in \{2, \ldots l\} \mid \beta(e_{C_j}) = 0\}$. If $C = \{s_1, \ldots, s_k\}$ where the s_j are pairwise $\approx_\beta$-equivalent and s^β_j is a β-term for s_j, we define

$$\mathcal{E}^\beta_C \;=\; \bigcup_{i \neq j} \mathcal{E}^{=_\beta}_{s^\beta_i, s^\beta_j}.$$

Note that $\beta(s^\beta_i) = \ulcorner\beta(s_i)\urcorner = \ulcorner\beta(s_j)\urcorner = \beta(s^\beta_j)$, and thus, $s^\beta_i =_\beta s^\beta_j$, and due to Lemma 5.5, $\mathcal{E}^{=_\beta}_{s^\beta_i, s^\beta_j}$ exists. Let

$$\begin{aligned} \text{If } J = \{2, \ldots, l\}, \text{ then : } & t^\beta = 1 \\ \text{otherwise : } & t^\beta = \Pi_{j \notin J \cup \{1\}} (s^\beta_{C_j})^{e_{C_j}} \end{aligned}$$

Furthermore, we define

$$\mathcal{E}^\beta_t \;=\; \bigcup_{i=1}^{n} \mathcal{E}^\beta_{t_i} \cup \bigcup_{j=2}^{l} \mathcal{E}^\beta_{C_j} \cup \bigcup_{j \in J} \{e_{C_j} = 0\}.$$

By induction, it is easy to see that $(t^\beta, \mathcal{E}^\beta_t)$ is a β-tuple for t: Induction yields that $\beta \models \mathcal{E}^\beta_t$. By the definition of the normalization function one easily verifies that if $J = \{2, \ldots, l\}$, then $\ulcorner\beta(t)\urcorner = 1$, and thus, $t^\beta = \ulcorner\beta(t)\urcorner$. Otherwise, it is easy to see that $\ulcorner\beta(t)\urcorner = \Pi_{j \notin J \cup \{1\}} \ulcorner\beta(s_{C_j})\urcorner^{\beta(e_{C_j})}$. By induction, $\beta(s^\beta_{C_j}) = \ulcorner\beta(s_{C_j})\urcorner$. Thus, $\beta(t^\beta) = \ulcorner\beta(t)\urcorner$. Now, let $\beta' \models \mathcal{E}^\beta_t$. Let $s, s' \in C_j$ with $s \neq s'$. By definition of $\mathcal{E}^\beta_t$

we have $\beta' \models \mathcal{E}_s^\beta \cup \mathcal{E}_{s'}^\beta \cup \mathcal{E}_{s,s'}^{=_\beta} (= \mathcal{E}_{s,s'}^{\approx_\beta})$. Thus, by Lemma 5.9, $\ulcorner \beta'(s) \urcorner = \ulcorner \beta'(s') \urcorner$. We also know $s^\beta = 1$, and thus, $\beta'(s^\beta) = 1$ for every $s \in C_1$. Finally, if $j \in J$, we have that $\beta'(e_{C_j}) = 0$. Now, it is easy to see that $\ulcorner \beta'(t) \urcorner = \ulcorner \beta'(t^\beta) \urcorner$.

— If $t = Exp(u, M)$ and $\ulcorner \beta(u) \urcorner \neq Exp(\cdot, \cdot)$, then by induction, there exists a β-tuple $(u^\beta, \mathcal{E}_u^\beta)$ for u and a β-tuple $(M^\beta, \mathcal{E}_M^\beta)$ for M. Let

$$\begin{array}{rl} \text{If } \ulcorner \beta(M) \urcorner = 1, \text{ then : } & t^\beta \ = \ u^\beta, \\ \text{otherwise : } & t^\beta \ = \ Exp(u^\beta, M^\beta) \end{array}$$

Furthermore, in both cases we set

$$\mathcal{E}_t^\beta \ = \ \mathcal{E}_u^\beta \cup \mathcal{E}_M^\beta.$$

We will conclude that $(t^\beta, \mathcal{E}_t^\beta)$ is a β-tuple for t by induction: Obviously, $\beta \models \mathcal{E}_t^\beta$. Therefore, we must show that $\beta(t^\beta) = \ulcorner \beta(t) \urcorner$ and $\ulcorner \beta'(t) \urcorner = \ulcorner \beta'(t^\beta) \urcorner$ for every $\beta' \models \mathcal{E}_t^\beta$. Two cases arise :

(1) Either $\ulcorner \beta(M) \urcorner = 1$, and thus, $\ulcorner \beta(t) \urcorner = \ulcorner \beta(u) \urcorner$. Thus, by induction, $\ulcorner \beta(t) \urcorner = \ulcorner \beta(u) \urcorner = \beta(u^\beta) = \beta(t^\beta)$. Also, we have that $\beta(M^\beta) = \ulcorner \beta(M) \urcorner = 1$, and thus, $M^\beta = 1$. By induction, $1 = \ulcorner \beta'(M^\beta) \urcorner = \ulcorner \beta'(M) \urcorner$. Hence,

$$\ulcorner \beta'(t) \urcorner \ = \ \ulcorner \beta'(u) \urcorner \ \overset{(*)}{=} \ \ulcorner \beta'(u^\beta) \urcorner \ \overset{(**)}{=} \ \ulcorner \beta'(t^\beta) \urcorner$$

where $(*)$ is by induction and definition of $\mathcal{E}_t^\beta$, and $(**)$ by definition of t.

(2) Or $\ulcorner \beta(M) \urcorner \neq 1$. Thus, $\ulcorner \beta(t) \urcorner = Exp(\ulcorner \beta(u) \urcorner, \ulcorner \beta(M) \urcorner) \overset{(*)}{=} Exp(\beta(u^\beta), \beta(M^\beta)) \overset{(**)}{=} \beta(t^\beta)$ where $(*)$ is by induction and $(**)$ is by definition of t. Now, let $\beta' \models \mathcal{E}_t^\beta$. We have

$$\begin{array}{rl} \ulcorner \beta'(t) \urcorner \ = & \ulcorner Exp(\ulcorner \beta'(u) \urcorner, \ulcorner \beta'(M) \urcorner) \urcorner \ \overset{(*)}{=} \ \ulcorner Exp(\ulcorner \beta'(u^\beta) \urcorner, \ulcorner \beta'(M^\beta) \urcorner) \urcorner \\ = & \ulcorner Exp(\beta'(u^\beta), \beta'(M^\beta)) \urcorner \ \overset{(**)}{=} \ \ulcorner \beta'(t^\beta) \urcorner \end{array}$$

where $(*)$ is by induction and definition of $\mathcal{E}_t^\beta$, and $(**)$ by definition of t^β.

— If $t = Exp(u, M)$ and $\ulcorner \beta(u) \urcorner = Exp(u', M')$, by induction there exists a β-tuple $(u^\beta, \mathcal{E}_u^\beta)$ for u. In particular, $\beta(u^\beta) = \ulcorner \beta(u) \urcorner$, and thus, u^β is of the form $Exp(u'', M'')$ where $\beta(u'') = u'$ and $\beta(M'') = M'$. Moreover, by induction there exists a β-tuple $((M'' \cdot M)^\beta, \mathcal{E}_{(M'' \cdot M)}^\beta)$ for $(M'' \cdot M)$. Let

$$\begin{array}{ll} t^\beta = u'' & \text{if } \ulcorner \beta(M'' \cdot M) \urcorner = 1, \text{ i.e. } \ulcorner M' \cdot \beta(M) \urcorner = 1 \\ t^\beta = Exp(u'', (M'' \cdot M)^\beta) & \text{otherwise.} \end{array}$$

In both cases, we set

$$\mathcal{E}_t^\beta \ = \ \mathcal{E}_u^\beta \cup \mathcal{E}_{(M'' \cdot M)}^\beta.$$

By induction, we can now show that $(t^\beta, \mathcal{E}_t^\beta)$ is a β-tuple for t: Obviously, $\beta \models \mathcal{E}_t^\beta$. Therefore, we must only show that $\beta(t^\beta) = \ulcorner \beta(t) \urcorner$ and $\ulcorner \beta'(t) \urcorner = \ulcorner \beta'(t^\beta) \urcorner$ for every $\beta' \models \mathcal{E}_t^\beta$. First, let $(u^\beta, \mathcal{E}_u^\beta) = Exp(u'', M'')$ as above. We know that $\ulcorner \beta(t) \urcorner = \ulcorner Exp(\ulcorner \beta(u) \urcorner, \beta(M)) \urcorner = \ulcorner Exp(u', \ulcorner M' \cdot \beta(M) \urcorner) \urcorner$. If $\ulcorner M' \cdot \beta(M) \urcorner = 1$, then $\ulcorner \beta(t) \urcorner = u' = \beta(u'') = \beta(t^\beta)$. Otherwise,

$$\begin{array}{rlll} \ulcorner \beta(t) \urcorner \ = & Exp(u', \ulcorner M' \cdot \beta(M) \urcorner) & = & Exp(\beta(u''), \ulcorner \beta(M'') \cdot \beta(M) \urcorner) \\ = & Exp(\beta(u''), \ulcorner \beta(M'' \cdot M) \urcorner) & \overset{(*)}{=} & Exp(\beta(u''), \beta((M'' \cdot M)^\beta)) \quad \overset{(**)}{=} \ \beta(t^\beta) \end{array}$$

where we obtain $(*)$ by induction and $(**)$ by definition of t^β. Now, let $\beta' \models \mathcal{E}_t^\beta$. We have that $\ulcorner\beta'(t)\urcorner = \ulcorner Exp(\beta'(u), \beta'(M))\urcorner$. Since $\beta' \models \mathcal{E}_u^\beta$, it follows by induction that $\ulcorner\beta'(u)\urcorner = \ulcorner\beta'(u^\beta)\urcorner = \ulcorner\beta'(Exp(u'', M''))\urcorner$. Thus, $\ulcorner\beta'(t)\urcorner = \ulcorner Exp(\beta'(u^\beta), \beta'(M))\urcorner = \ulcorner Exp(\beta'(u''), \beta'(M'') \cdot \beta'(M))\urcorner$, and therefore $\ulcorner\beta'(t)\urcorner = \ulcorner Exp(\beta'(u''), \beta'(M'' \cdot M))\urcorner$. Moreover, since $\beta' \models \mathcal{E}^\beta_{(M''\cdot M)}$, induction yields $\ulcorner\beta'((M'' \cdot M)^\beta)\urcorner = \ulcorner\beta'(M'' \cdot M)\urcorner$. Consequently, $\ulcorner\beta'(t)\urcorner = \ulcorner Exp(\beta'(u''), \beta'((M'' \cdot M)^\beta))\urcorner$. If $t^\beta = Exp(u'', (M'' \cdot M)^\beta)$, then we obtain $\ulcorner\beta'(t)\urcorner = \ulcorner t^\beta\urcorner$. Otherwise, $t^\beta = u''$ and $\ulcorner\beta(M'' \cdot M)\urcorner = 1$, and thus, $(M'' \cdot M)^\beta = 1$. Hence, $\ulcorner\beta'(t)\urcorner = \ulcorner t^\beta\urcorner$. □

5.5 Bounding the size of β-terms

From now on, we will denote by t^β the β-term of t as constructed in the proof of Lemma 5.10. We want to show that there always exists a β-tuple of t of size polynomially bounded in $||t||$. This is done in two steps: In this subsection, we first bound the size of t^β. Then, in the next subsection, we bound the size of the equation system associated to t^β.

First, we need to prove that β-terms are uniquely determined:

LEMMA 5.11. *For every open message or product t such that $\beta(t) = \ulcorner\beta(t)\urcorner$, we have that $t^\beta = t$.*

PROOF. See Appendix 9.3. □

For a set E of open messages or products, we define $E^\beta = \{t^\beta \mid t \in E\}$. In the following lemma we bound $|t^\beta|_{ext}$ and in Lemma 5.14 we bound $||t^\beta||_{exp}$. Both lemmas are put together in Lemma 5.15 to yield a bound for $||t^\beta||_{ext}$.

LEMMA 5.12. *For open messages and products $t, t_1, \ldots, t_n$ and an evaluation mapping β we have that*

(1) $\mathcal{S}(t^\beta) \subseteq \mathcal{S}(t)^\beta$.

(2) $|\ulcorner\beta(t)\urcorner| \leq |t|$ *and* $|\ulcorner\beta(t_1)\urcorner, \ldots, \ulcorner\beta(t_1)\urcorner| \leq |t_1, \ldots, t_n|$,

(3) $|t^\beta|_{ext} \leq 2 \cdot |t|$.

PROOF. We prove these statements one by one:

(1): We proceed by structural induction on t.

—If $t \in \mathcal{A}$, we have $\mathcal{S}(t^\beta) = \{t\} = \mathcal{S}(t)^\beta$. If $t = \langle t_1, t_2\rangle$, induction yields that $\mathcal{S}(t^\beta) = \{t^\beta\} \cup \mathcal{S}(t_1^\beta) \cup \mathcal{S}(t_2^\beta) \subseteq \{t^\beta\} \cup \mathcal{S}(t_1)^\beta \cup \mathcal{S}(t_2)^\beta = \mathcal{S}(t)^\beta$; analogously for encryption.

—If $t = t_1^{e_1} \cdots t_n^{e_n}$, we have two cases: Either $t^\beta = 1$, and we obviously have $\mathcal{S}(t^\beta) \subseteq \mathcal{S}(t)^\beta$. Or $t^\beta = \Pi_{j \notin J \cup \{1\}}(s^\beta_{C_j})^{e_{C_j}}$ (see the proof of Lemma 5.10) where for each $s^\beta_{C_j}$ there exists a t_i such that $t_i^\beta = s^\beta_{C_j}$. Thus, induction yields that :

$$\begin{array}{ll} \mathcal{S}(t^\beta) & \subseteq \{t^\beta\} \cup \bigcup_{j \notin J \cup \{1\}} \mathcal{S}(s^\beta_{C_j}) \subseteq \{t^\beta\} \cup \bigcup_{i=1}^n \mathcal{S}(t_i^\beta) \\ & \subseteq \{t^\beta\} \cup \bigcup_{i=1}^n \mathcal{S}(t_i)^\beta \qquad = \mathcal{S}(t)^\beta \end{array}$$

—If $t = Exp(u, M)$ and $t^\beta = u^\beta$, induction immediately yields $\mathcal{S}(t^\beta) \subseteq \mathcal{S}(t)^\beta$.

—If $t = Exp(u, M)$, $t^\beta = Exp(u^\beta, M^\beta)$, and $M = t_1^{e_1} \cdots t_n^{e_n}$, then $M^\beta \neq 1$ and, by induction,

$$\begin{aligned}\mathcal{S}(t^\beta) &\subseteq \{t^\beta\} \cup \mathcal{S}(u^\beta) \cup \textstyle\bigcup_i \mathcal{S}(t_i^\beta) \\ &\subseteq \{t^\beta\} \cup \mathcal{S}(u)^\beta \cup \textstyle\bigcup_i \mathcal{S}(t_i)^\beta \ = \mathcal{S}(t)^\beta\end{aligned}$$

—If $t = Exp(u, M)$, $u^\beta = Exp(u'', M'')$, and $t^\beta = u''$. Then, $\mathcal{S}(t^\beta) \subseteq \mathcal{S}(u^\beta) \subseteq \mathcal{S}(u)^\beta \subseteq \mathcal{S}(t)^\beta$.

—Finally, assume that $t = Exp(u, M)$, $u^\beta = Exp(u'', M'')$, and $t^\beta = Exp(u'', (M'' \cdot M)^\beta)$ where $(M'' \cdot M)^\beta \neq 1$. Thus, $(M'' \cdot M)^\beta$ is of the form $\bigcup_{j \notin J \cup \{1\}} (s_{C_j}^\beta)^{e'_j}$ for some e'_j such that with $M'' = t''^{e''_1}_1 \cdots t''^{e''_{n''}}_{n''}$ and $M = t_1^{e_1} \cdots t_n^{e_n}$ every $s_{C_j}^\beta$ equals some t''_i or t_i^β. (Note that $t''_i = t''^\beta_i$ by Lemma 5.11.) By induction, $t''_i \in \mathcal{S}(u^\beta) \subseteq \mathcal{S}(u)^\beta$. We conclude that

$$\begin{aligned}\mathcal{S}(t^\beta) &= \{t^\beta\} \cup \mathcal{S}(u'') \cup \textstyle\bigcup_{j \notin J \cup \{1\}} \mathcal{S}(s_{C_j}^\beta) \\ &\subseteq \{t^\beta\} \cup \mathcal{S}(u'') \cup \textstyle\bigcup_{i=1}^{n} \mathcal{S}(t_i^\beta) \cup \bigcup_{i=1}^{n''} \mathcal{S}(t''_i) \\ &\subseteq \{t^\beta\} \cup \textstyle\bigcup_{i=1}^{n} \mathcal{S}(t_i)^\beta \cup \mathcal{S}(u^\beta) \\ &\subseteq \{t^\beta\} \cup \textstyle\bigcup_{i=1}^{n} \mathcal{S}(t_i)^\beta \cup \mathcal{S}(u)^\beta = \mathcal{S}(t)^\beta\end{aligned}$$

(2): First note that $\ulcorner \beta(t) \urcorner = \beta(t^\beta)$. It is easy to see that $|\beta(s)| \leq |s|$ for every open message and product s. Thus,

$$|\ulcorner \beta(t) \urcorner| = |\beta(t^\beta)| \leq |t^\beta| \overset{(*)}{\leq} \mathsf{Card}(\mathcal{S}(t)^\beta) \leq \mathsf{Card}(\mathcal{S}(t)) = |t|$$

where for $(*)$ we use 1. The same argument works for $|\ulcorner \beta(t_1) \urcorner, \ldots, \ulcorner \beta(t_1) \urcorner| \leq |t_1, \ldots, t_n|$.

(3): This is an immediate consequence of 1. and Lemma 5.2: $|t^\beta|_{ext} \leq 2 \cdot |t^\beta| \leq 2 \cdot \mathsf{Card}(\mathcal{S}(t)^\beta) \leq 2 \cdot |t|$. □

We now want to bound $||t^\beta||_{exp}$. To do this, we need the following lemma:

Lemma 5.13. *Let E be a finite set of open messages or products such that $\mathcal{S}_{ext}(E) = E$ and t maximal (w.r.t. subterm ordering) in E. Then :*

$$|\bigcup_{s \in E} \mathcal{S}_{ext}(s^\beta)|_{exp} \leq |\bigcup_{s \in E \setminus \{t\}} \mathcal{S}_{ext}(s^\beta)|_{exp} + ||t||^2_{ext}$$

Proof. See Appendix 9.3. □

We use this lemma to prove:

Lemma 5.14. *For every open message or product t, it follows that $||t^\beta||_{exp} \leq ||t||^3_{ext}$.*

Proof. Let $E = \mathcal{S}_{ext}(t)$. Thanks to Lemma 5.13, we know that $||t^\beta||_{exp} = |\mathcal{S}_{ext}(t^\beta)|_{exp} \leq |\bigcup_{s \in E} \mathcal{S}_{ext}(s^\beta)|_{exp}$. This allows us to iteratively extract a (maximal) term from E and shows that $|\bigcup_{s \in E} \mathcal{S}_{ext}(s^\beta)|_{exp} \leq |t|_{ext} \cdot ||t||^2_{ext}$. (Note that $\mathsf{Card}(E) = |t|_{ext}$.) This yields $||t^\beta||_{exp} \leq ||t||^3_{ext}$. □

We have bounded both the number of extended subterms of t^β and the size of its integer coefficients (Lemma 5.12 and 5.14). Putting this together, we finally obtain the polynomial bound on β-terms:

LEMMA 5.15. *For every open message or product t, we have $||t^\beta||_{ext} \leq 3 \cdot ||t||^3_{ext}$.*

5.6 Bounding the size of β-equation systems

In the previous subsection, we have bounded the size of β-terms t. We now bound the size of β-equation systems associated to these terms, and hence, bound the size of β-tuples. To do so, we first construct a particular β-equation system for t whose size can be polynomially bounded in $||t||_{ext}$.

In what follows, β is an evaluation mapping and t an open message or product. We first describe the equation system ${\mathcal{E}'}^\beta_t$ for an open message or product t added when going from the equation system of the subterms of t to that of t:

—If t is atomic, a pair, or encryption, then ${\mathcal{E}'}^\beta_t = \emptyset$.

—If $t = t_1^{e_1} \cdots t_n^{e_n}$, then with the notation used in the proof of Lemma 5.10, we set ${\mathcal{E}'}^\beta_t = \bigcup_{j=2}^{l} \mathcal{E}^\beta_{C_j} \cup \bigcup_{j \in J} \{e_{C_j} = 0\}$.

—If $t = Exp(u, M)$ and $\ulcorner \beta(u) \urcorner \neq Exp(\cdot, \cdot)$, then ${\mathcal{E}'}^\beta_t = \emptyset$. Otherwise, $u^\beta = Exp(u'', M'')$ and we set ${\mathcal{E}'}^\beta_t = {\mathcal{E}'}^\beta_{(M'' \cdot M)}$.

Remark 5.16. The equation system ${\mathcal{E}'}^\beta_t$ is uniquely determined (modulo AC_+).

The equation system ${\mathcal{E}'}^\beta_t$ describes the constraints on product exponents on one level of t. The complete equation system for t is defined as the union of the equation systems for all subterms of t:

$$\mathcal{E}^\beta_t = \bigcup_{s \in \mathcal{S}_{ext}(t)} {\mathcal{E}'}^\beta_s.$$

We now prove that $(t^\beta, \mathcal{E}^\beta_t)$ is a β-tuple for t and that the size of $\mathcal{E}^\beta_t$ is polynomially bounded in the size of t. But first, we need to show the following lemma for β-tuples on products:

LEMMA 5.17. *Let $M = t_1^{e_1} \cdots t_n^{e_n}$ and $M' = {t'}_1^{e'_1} \cdots {t'}_{n'}^{e'_{n'}}$ such that $\beta(t'_i) = \ulcorner \beta(t'_i) \urcorner$. Let $(t_i^\beta, \mathcal{E}_i)$ be a β-tuple for t_i for every i. Then, $((M' \cdot M)^\beta, {\mathcal{E}'}^\beta_{(M' \cdot M)} \cup \bigcup_i \mathcal{E}_i)$ is a β-tuple for $M' \cdot M$.*

PROOF. See Appendix 9.3. □

LEMMA 5.18. *For every open message or product t and every evaluation mapping β the following is true.*

(1) The tuple $(t^\beta, \mathcal{E}^\beta_t)$ is a β-tuple for t.

(2) The size of ${\mathcal{E}'}^\beta_t$ is bounded by a polynomial in $||t||_{ext}$.

(3) The size of $(t^\beta, \mathcal{E}^\beta_t)$, which is the some of the size of t^β and $\mathcal{E}^\beta_t$ is bounded by a polynomial in $||t||_{ext}$.

PROOF. We prove these statements one by one.

(1): We proceed by structural induction on t according to the construction in Lemma 5.10.

—The case where $t \in \mathcal{A}$ is obvious. Now, assume that $t = \langle t_1, t_2 \rangle$. Then, $\mathcal{E}_t^\beta = \mathcal{E}'^\beta_t \cup \bigcup_{s \in \mathcal{S}_{ext}(t_1)} \mathcal{E}'^\beta_s \cup \bigcup_{s \in \mathcal{S}_{ext}(t_2)} \mathcal{E}'^\beta_s$. By definition, $\mathcal{E}_t^\beta = \mathcal{E}_{t_1}^\beta \cup \mathcal{E}_{t_2}^\beta$. (Note that here we use Remark 5.16.) Just as in the proof of Lemma 5.10, from this we can conclude that $(t^\beta, \mathcal{E}_t^\beta)$ is a β-tuple. The argument for encryption is similar.

—If $t = t_1^{e_1} \cdots t_n^{e_n}$ and with the notation introduced in the proof of Lemma 5.10, by definition of $\mathcal{E}'^\beta_t$ and using Remark 5.16, we can conclude that $\mathcal{E}_t^\beta = \bigcup_i \mathcal{E}_{t_i}^\beta \cup \bigcup_{j=2}^l \mathcal{E}_{C_j}^\beta \cup \bigcup_{j \in J} \{e_{C_j} = 0\}$. Again, as in the proof of Lemma 5.10 it follows that $(t^\beta, \mathcal{E}_t^\beta)$ is a β-tuple for t.

—If $t = \mathit{Exp}(u, M)$ and $\ulcorner \beta(u) \urcorner \neq \mathit{Exp}(\cdot, \cdot)$, then by definition of $\mathcal{E}_t^\beta$ and using Remark 5.16 we have $\mathcal{E}_t^\beta = \mathcal{E}_u^\beta \cup \mathcal{E}_M^\beta$ and as in Lemma 5.10 this implies that $(t^\beta, \mathcal{E}_t^\beta)$ is a β-tuple for t.

—Finally, assume that $t = \mathit{Exp}(u, M)$, $\ulcorner \beta(u) \urcorner \neq \mathit{Exp}(\cdot, \cdot)$, and $u^\beta = \mathit{Exp}(u'', M'')$. Assume that $M = t_1^{e_1} \cdots t_n^{e_n}$. Lemma 5.17 implies that $((M'' \cdot M)^\beta, \mathcal{E}'^\beta_{(M'' \cdot M)} \cup \bigcup_i \mathcal{E}_{t_i}^\beta)$ is a β-tuple for $M'' \cdot M$. By definition of $\mathcal{E}_t^\beta$ and Remark 5.16 we have that $\mathcal{E}_u^\beta \cup \bigcup_i \mathcal{E}_{t_i}^\beta \cup \mathcal{E}'^\beta_{(M'' \cdot M)} \subseteq \mathcal{E}_t^\beta$. Then, as in the proof of Lemma 5.10 it follows that $(t^\beta, \mathcal{E}_t^\beta)$ is a β-tuple for t.

(2): In case t is atomic, a pair, or encryption, nothing is to show. In case t is a product, using Lemma 5.5 and 5.15 it is easy to see that $\mathcal{E}'^\beta_t$ can be bounded by a polynomial in $||t||_{ext}$. For the case $t = \mathit{Exp}(\cdot, \cdot)$, one obtains a polynomial in $||t||_{ext}$ bounding the size of $\mathcal{E}'^\beta_t$ using Lemma 5.15 and the case where t is a product.

(3): If p is the polynomial bounding the size of $\mathcal{E}'^\beta_t$, then $p(||t||_{ext}) \cdot ||t||_{ext}$ bounds the size of $\mathcal{E}_t^\beta$. By Lemma 5.15 we know that the size of t^β can be bounded by a polynomial in $||t||_{ext}$. □

Finally, by Lemma 5.5, 5.9, and 5.18 we have established the existence of particular $\approx_\beta$-equation systems whose size is polynomially bounded:

PROPOSITION 5.19. *Let t and t' be open messages or open products and β be an evaluation mapping such that $t \approx_\beta t'$. Then, there exists a $\approx_\beta$-equation system for t and t' of size polynomially bounded in $||t, t'||_{ext}$.*

We will denote such an equation system by $\mathcal{E}_{t,t'}^{\approx_\beta}$.

5.7 Bounding the Size of Product Exponents in Attacks

We are now prepared to prove the key lemma of this section, Lemma 5.21. As mentioned, using this lemma, we can conclude that DH rules allow polynomial product exponent attacks (Proposition 5.22). As an immediate consequence, we obtain that INSECURE is NP-complete for the DH intruder (Theorem 5.23).

Lemma 5.21 is proved in two steps. First, a restricted version is considered where only one intruder rule is applied (Lemma 5.20). Then, this is extended to complete derivations.

In the following proofs, we will consider extensions of evaluation mappings. We say that a mapping $\beta' : Z' \to \mathbb{Z}$ is an *extension* of an evaluation mapping $\beta : Z \to \mathbb{Z}$

if $Z \subseteq Z'$ and $\beta'(z) = \beta(z)$ for all $z \in Z$. Since β and β' coincide on Z, by abuse of notation, we often refer to an extension of β by β.

LEMMA 5.20. *Let $t_1, \ldots, t_n$ be open messages, s be a normalized message, β be an evaluation mapping, and $L \in \mathcal{L}$ an intruder rule such that $\beta(t_i) = \lceil \beta(t_i) \rceil$ for all i and $\lceil \beta(t_1) \rceil, \ldots, \lceil \beta(t_n) \rceil \to s \in L$. Then, there exists an open message t, an equation system $\mathcal{E}$, and an extension β of β such that*

(1) $\beta(t) = s$,
(2) $\beta \models \mathcal{E}$,
(3) $\lceil \beta'(t_1) \rceil \ldots, \lceil \beta'(t_n) \rceil \to \lceil \beta'(t) \rceil \in \mathcal{L}$ for every $\beta' \models \mathcal{E}$,
(4) $\max\{|e| \mid e \in \mathcal{L}_{exp}(t)\} \leq \max\{|e| \mid e \in \mathcal{L}_{exp}(t_1, \ldots, t_n)\} + n$, and
(5) the size of $\mathcal{E}$ is polynomially bounded in $||t_1, \ldots, t_n||_{ext}$.

PROOF. See Appendix 9.3. □

We will now extend this lemma to complete derivations. In Section 5.2, we have provided some intuition behind the proof of this lemma.

LEMMA 5.21. *Let $t, t_1, \ldots, t_n$ be open messages such that there exists a derivation witnessing $\lceil \beta(t) \rceil \in \mathit{forge}(\lceil \beta(t_1) \rceil, \ldots, \lceil \beta(t_n) \rceil)$. Then, there exists an extension of β and an equation system $\mathcal{E}$ such that*

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

PROOF. Let $E = \{\lceil \beta(t_1) \rceil, \ldots, \lceil \beta(t_n) \rceil\}$ and let D be a well-formed derivation witnessing $\lceil \beta(t) \rceil \in \mathit{forge}(E)$. We know that the length l of D is polynomially bounded in $|\lceil \beta(t_1) \rceil, \ldots, \lceil \beta(t_n) \rceil, \lceil \beta(t) \rceil|$. By Lemma 5.12, 2. $|\lceil \beta(t_1) \rceil, \ldots, \lceil \beta(t_n) \rceil, \lceil \beta(t) \rceil|$ is bounded by a polynomial in $|t_1, \ldots, t_n, t|$, and thus, in $||t_1, \ldots, t_n, t||_{ext}$. Assume that the ith step of D is $E, s_1, \ldots, s_{i-1} \to_{L_i} E, s_1, \ldots, s_i$ for every $1 \leq i \leq l$ where s_i is a normalized message for every i and $s_l = \lceil \beta(t) \rceil$. Since D is well-formed we have that $s_i \in \mathcal{S}(\lceil \beta(t_1) \rceil, \ldots, \lceil \beta(t_n) \rceil, \lceil \beta(t) \rceil)$ for every i.

Let $(t^\beta, \mathcal{E}_t^\beta)$ be a β-tuple of t and t_i^β be a β-tuple of t_i for every i. It follows that $\beta(t_i^\beta) = \lceil \beta(t_i) \rceil$, and thus, $E = \{\beta(t_1^\beta), \ldots, \beta(t_n^\beta)\}$. Hence, to the first step of D we can apply Lemma 5.20 and obtain an open message s_1', an equation system $\mathcal{E}_1$, and an extension of β such that $\beta(s_1') = s_1$, $\beta \models \mathcal{E}_1$, and $\lceil \beta'(t_1^\beta) \rceil, \ldots, \lceil \beta'(t_n^\beta) \rceil \to_{L_1} \lceil \beta'(t_1^\beta) \rceil, \ldots, \lceil \beta'(t_n^\beta) \rceil, \lceil \beta'(s_1') \rceil$ for every $\beta' \models \mathcal{E}_1$.

Note that $\beta(t_i^\beta) = \lceil \beta(t_i^\beta) \rceil$ for every i and $\beta(s_1') = \lceil \beta(s_1') \rceil = s_1$. Thus, we can apply Lemma 5.20 inductively and obtain s_j', $\mathcal{E}_j$, and an extension of β such that $\beta(s_j') = s_j$, $\beta \models \mathcal{E}_j$ and $\lceil \beta'(t_1^\beta) \rceil, \ldots, \lceil \beta'(t_n^\beta) \rceil, \lceil \beta'(s_1') \rceil, \ldots, \lceil \beta'(s_{j-1}') \rceil \to_{L_j} \lceil \beta'(t_1^\beta) \rceil, \ldots, \lceil \beta'(t_n^\beta) \rceil, \lceil \beta'(s_1') \rceil, \ldots, \lceil \beta'(s_j') \rceil$ for every $\beta' \models \mathcal{E}_j$ and $1 \leq j \leq l$. Consequently, $\beta \models \bigcup_{j=1}^{l} \mathcal{E}_j$ and $\lceil \beta'(s_l') \rceil \in \mathit{forge}(\lceil \beta'(t_1^\beta) \rceil, \ldots, \lceil \beta'(t_n^\beta) \rceil)$ for every $\beta' \models \bigcup_{j=1}^{l} \mathcal{E}_j$.

If $\beta' \models \bigcup_{i=1}^{n} \mathcal{E}_{t_i}^\beta$, then $\lceil \beta'(t_i) \rceil = \lceil \beta'(t_i^\beta) \rceil$. We know that $\beta(t^\beta) = \lceil \beta(t) \rceil = s_l = \beta(s_l')$. Thus, $t^\beta =_\beta s_l'$. Consequently, due to Lemma 5.5, the $=_\beta$-equation system

$\mathcal{E}^{=_\beta}_{t^\beta,s'_l}$ for t^β and s'_l exists. Now, if $\beta' \models \mathcal{E}^\beta_t \cup \mathcal{E}^{=_\beta}_{t^\beta,s'_l}$ we obtain $\ulcorner\beta'(t)\urcorner = \ulcorner\beta'(t^\beta)\urcorner = \ulcorner\beta'(s'_l)\urcorner$. We set

$$\mathcal{E} = \bigcup_{i=1}^{n} \mathcal{E}^\beta_{t_i} \cup \mathcal{E}^\beta_t \cup \bigcup_{j=1}^{l} \mathcal{E}_j \cup \mathcal{E}^{=_\beta}_{t^\beta,s'_l}.$$

It follows that $\beta \models \mathcal{E}$ and $\ulcorner\beta'(t)\urcorner \in \mathit{forge}(\ulcorner\beta'(t_1)\urcorner, \ldots, \ulcorner\beta'(t_n)\urcorner)$ for every $\beta' \models \mathcal{E}$.

It remains to show that $\mathcal{E}$ is polynomially bounded in $||t_1, \ldots, t_n, t||_{ext}$. Lemma 5.20 implies that every $\mathcal{E}_j$ is polynomially bounded in $||t_1, \ldots, t_n, s'_1, \ldots, s'_{j-1}||_{ext}$. We have that $\beta(s'_j) = s_j \in \mathcal{S}(\ulcorner\beta(t_1)\urcorner, \ldots, \ulcorner\beta(t_n)\urcorner, \ulcorner\beta(t)\urcorner)$. Thus, using Lemma 5.12, $|s'_j|$ can polynomially be bounded in $|t_1, \ldots, t_n, t|$. By Lemma 5.2, $|s'_j|_{ext}$ can polynomially be bounded in $|t_1, \ldots, t_n, t|$. From Lemma 5.20 it follows that $\max\{|e| \mid e \in \mathcal{L}_{exp}(s'_j)\} \leq \max\{|e| \mid e \in \mathcal{L}_{exp}(t^\beta_1, \ldots, t^\beta_n)\} + n \cdot (j-1)$. We know that $j \leq l$ and that l is polynomially bounded in $||t_1, \ldots, t_n, t||_{ext}$. Also, $\max\{|e| \mid e \in \mathcal{L}_{exp}(t^\beta_1, \ldots, t^\beta_n)\} \leq ||t^\beta_i||_{exp} \leq ||t_i||^3_{ext}$ for some i (see Lemma 5.14). Thus, there exists a polynomial p such that $||s'_j||_{ext}$ is bounded by $p(||t_1, \ldots, t_n, t||_{ext})$. Consequently, $||t_1, \ldots, t_n, t, s'_1, \ldots, s'_j||_{ext}$ is bounded by $(p'(||t_1, \ldots, t_n, t||_{ext}) + 1) \times p(||t_1, \ldots, t_n, t||_{ext})$ where p' is the polynomial bounding l. By Lemma 5.20, this shows that $\mathcal{E}_j$ is polynomially bounded in $||t_1, \ldots, t_n, t||_{ext}$. Lemma 5.18 and 5.5 now imply that $\mathcal{E}$ is polynomially bounded in $||t_1, \ldots, t_n, t||_{ext}$. □

We can now show that DH rules allow polynomial product exponent attacks.

PROPOSITION 5.22. *DH rules allow polynomial product exponent attacks.*

PROOF. Let (π, σ) be a minimal attack on P. 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. Thus, $\sigma(x) = \beta(\sigma^Z(x))$ for every $x \in \mathcal{V}(P)$. Note that, due to Corollary 3.16, $|\sigma^Z|$ can polynomially be bounded in $|P|$. Since the product exponents in σ^Z are variables, we have that $||\sigma^Z||_{exp} \leq |\sigma^Z|^2$. Thus, $||\sigma^Z||_{ext}$ can polynomially be bounded in $||P||_{ext}$.

Let k, $R_1, \ldots, R_k$, $S_0, \ldots, S_k$ be defined as usual. W.l.o.g. we assume that S_0 is a single message instead of a set of messages. (Otherwise, represent $S_0 = \{a_1, \ldots, a_n\}$ by the term $\langle a_1, \langle a_2 \ldots \langle a_{n-1}, a_n\rangle \cdots\rangle\rangle$.)

Let $R_{k+1} = \mathsf{secret}$. We know that $\ulcorner\beta(R_i\sigma^Z)\urcorner \in \mathit{forge}(\ulcorner\beta(S_0\sigma^Z)\urcorner, \ldots, \ulcorner\beta(S_{i-1}\sigma^Z)\urcorner$, for every $1 \leq i \leq k+1$.

By Lemma 5.21, for every i there exists an extension of β (the different extensions are independent from each other) and an equation system $\mathcal{E}_i$ such that

—$\beta \models \mathcal{E}_i$,
—$\ulcorner\beta'(R_i\sigma^Z)\urcorner \in \mathit{forge}(\ulcorner\beta'(S_0\sigma^Z)\urcorner, \ldots, \ulcorner\beta'(S_{i-1}\sigma^Z)\urcorner$ for every $\beta' \models \mathcal{E}_i$, and
—the size of $\mathcal{E}_i$ is polynomially bounded in $||S_0\sigma^Z, .., S_k\sigma^Z, R_1\sigma^Z, .., R_{k+1}\sigma^Z||_{ext}$ which in turn can polynomially be bounded in $||P||_{ext}$.

Consequently, $\beta \models \bigcup_{i=1}^k \mathcal{E}_i =: \mathcal{E}$, and thus, $\mathcal{E}$ is solvable, and for every $\beta' \models \mathcal{E}$ we have that $(\pi, \beta'(\sigma^Z))$ is an attack on P. By [Bockmayr and Weispfenning 2001], there exists a solution β' of $\mathcal{E}$ where the binary representation of the integers can polynomially be bounded in the size of $\mathcal{E}$, and thus, by Lemma 5.21, polynomially be bounded in $||P||_{ext}$. We define σ' to be $\beta'(\sigma^Z)$. Then, (π, σ') is an attack on

P. Also, $\sigma \approx \sigma'$, i.e., σ and σ' only different in the product exponents, and the $||\sigma'||_{exp}$ is polynomially bounded in $||P||_{ext}$, and thus, by Lemma 5.2, in $||P||$. □

In [Rusinowitch and Turuani 2001] it was shown that INSECURE is NP-hard in presence of the DY intruder. The proof easily carries over to the DH intruder. As an immediate consequence of Proposition 4.7, Proposition 4.8, Proposition 5.22, and Theorem 3.1 we obtain:

THEOREM 5.23. *The problem* INSECURE *is NP-complete for the DH intruder.*

6. THE A-GDH.2 PROTOCOL

The A-GDH.2 protocol [Steiner et al. 1998] allows a group of people who share pairwise long-term keys to establish a shared secret key using Diffie-Hellman exponentiation. We refer the reader to [Steiner et al. 1998] and [Pereira and Quisquater 2001] for a more detailed description of this 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:

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

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

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

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

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

Note that $Exp(y^{p,j}, r^{p,j} \cdot K_{p,p'}^{-1})$ 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 $M_{p_1 \cdots p_h}^{p,j}$ 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 $M_{p_1 \cdots p_h}^{p,j}$ to be

$$z_1^{p,j}, \ldots, z_{h+1}^{p,j} \Rightarrow Exp(z_1^{p,j}, r^{p,j} \cdot K_{p_1,p}), \ldots, Exp(z_h^{p,j}, r^{p,j} \cdot K_{p_h,p}), \{\mathsf{secret}^{p,j}\}^s_{Exp(z_{h+1}^{p,j}, r^{p,j})}$$

where the $z_k^{p,j}$ are variables, $Exp(z_k^{p,j}, r^{p,j} \cdot K_{p_k,p})$ is the keying material for p_k, and the message $Exp(z_{h+1}^{p,j}, r^{p,j})$ 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 $\Pi_{1,1,p''}^{p,1}, \Pi_{1,2,p''}^{p,1}$ (note that $\Pi_{1,1,p''}^{p,1} < \Pi_{1,2,p''}^{p,1}$), the rules for p' in the first session $\Pi_{2,1,p''}^{p',1}, \Pi_{2,2,p''}^{p',1}$, and the master $M_{pp'I}^{p'',1}$ of the first session. The protocol rules of the second session are $\Pi_{1,1,p''}^{p,2}, \Pi_{1,2,p''}^{p,2}, \Pi_{2,1,p''}^{p',2}, \Pi_{2,2,p''}^{p',2}, M_{pp'}^{p'',2}$. 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 [Pereira and Quisquater 2001], there exists an attack on P. It is easy to verify that this attack will be found by our decision procedure.

7. TRANSFERRING THE RESULTS TO COMMUTATIVE PUBLIC-KEY ENCRYPTION

In this section, we transfer the results obtained in Section 4 and 5 for Diffie-Hellman exponentiation to commutative public-key encryption (such as RSA with common modulus). We show that the insecurity problem is still NP-complete and that the derivation problem, i.e., the problem of deciding whether a given message can be derived from the finite set of messages, can be decided efficiently. These results can be obtained by slight modifications of the models and proofs presented in previous sections. This is possible since Diffie-Hellman exponentiation and commutative public-key encryption, which in case of RSA also involves exponentiation, share algebraic properties. Basically, we will now interpret the exponentiation operation $c = Exp(m, k_A)$ as the message m encrypted by (the public key) k_A where k'_A is the corresponding private key. By computing $Exp(c, k'_A) = Exp(m, k_A \cdot k'_A) = Exp(m, 1) = m$ the cipher c can be decrypted and the result is the plain-text m. We have commutativity of encryption as $Exp(Exp(m, k_A), k_B)$ and $Exp(Exp(m, k_B), k_A)$ are equivalent modulo the algebraic properties that we consider. Due to this new interpretation of exponentiation as public-key encryption some differences arise.

First, the intruder capabilities differ. In case of commutative public-key encryption the intruder is not able to compute the inverse of exponents, e.g., given a public key (n, e) and a cipher text $c = m^e \bmod n$, the intruder can not compute the private key d and then by computing $c^d \bmod n$ obtain m. Conversely, in the Diffie-Hellman setting, exponentiation is done modulo a publicly known prime, and

thus, it is computationally feasible to compute the inverse of exponents, e.g., given $m = g^{a \cdot b}$ and b where g generates the multiplicative group induced by the prime p, the intruder can easily compute the inverse b^{-1} of b modulo $p-1$ (in case an inverse exists) and by computing $m^{b^{-1}}$ obtain g^a.

Second, in the setting discussed so far, the intruder does not explicitly have inverses of messages in his knowledge, e.g., b^{-1}, since he can only have standard messages in his knowledge. However, in the public-key setting we consider in this section, this is too restrictive since inverses correspond to private keys, and of course, we need to allow the intruder to possess such keys (own private keys and private keys of dishonest principals).

In what follows, we will first provide two simple examples to illustrate the use of commutative public-key encryption systems in cryptographic protocols. We then indicate the changes necessary in our protocol and intruder model and finally state the main results of this section.

7.1 Examples of Protocols Relying on Commutative Public-key Encryption

The following two example protocols are taken from [Schneier 1996].

The first protocol is due to Shamir. The aim of this protocol is to permit secure communication between two agents who neither share a symmetric key nor know the public key of the other agent. The protocol uses the commutativity property of the RSA encryption system:

$$\begin{array}{ll} 1.\ A \to B: & \mathit{Exp}(\mathsf{secret}, K_A) \\ 2.\ B \to A: & \mathit{Exp}(\mathit{Exp}(\mathsf{secret}, K_A), K_B) \\ 3.\ A \to B: & \mathit{Exp}(\mathsf{secret}, K_B) \end{array}$$

In this protocol, a common RSA modulus n is assumed. The public key of A is (n, K_A) and the one for B is (n, K_B). The message secret is some non-negative integer $< n$. The term $\mathit{Exp}(\mathsf{secret}, K_A)$ stands for $\mathsf{secret}^{K_A} \bmod n$. By the algebraic properties of exponentiation, we have that $\mathit{Exp}(\mathit{Exp}(\mathsf{secret}, K_A), K_B) = \mathit{Exp}(\mathsf{secret}, K_A \cdot K_B) = \mathit{Exp}(\mathit{Exp}(\mathsf{secret}, K_B), K_A)$. In step 3 of the protocol, A computes $\mathit{Exp}(\mathit{Exp}(\mathit{Exp}(\mathsf{secret}, K_A), K_B), K'_A) = \mathit{Exp}(\mathsf{secret}, K_A \cdot K_B \cdot K'_A) = \mathit{Exp}(\mathsf{secret}, K_B)$ where K'_A is A's private key. Thus, the protocol itself uses the commutativity of encryption. Since B is not authenticated in this protocol, it is obvious that the intruder I can impersonate B, by simply playing B's role while using her own public key K_I.

A commutative public-key encryption system or signature scheme may also be relevant in the case of group protocols. Inspired by the protocol given in [Schneier 1996], Chapter 23, consider a group of l agents. A trusted server generates two large prime numbers p and q, computes $n = p \cdot q$, and $l+1$ numbers $k_0, \ldots, k_l$ such that:

$$k_0 \cdots k_l \equiv 1 \bmod (p-1) \cdot (q-1)$$

Each agent A_i, $1 \leq i \leq l$, receives for every j the public keys K_j which is the product of all k_0 to k_l except k_j and the private key k_i. Note that

$$\mathit{Exp}(M, k_0 \cdots k_l) = M,$$

and in particular,

$$Exp(Exp(M, k_i), K_i) = Exp(M, k_i \cdot K_i) = M.$$

Once the key distribution is completed, a message can be signed by a subset $\{A_i\}_{i \in I,\, I \subseteq [1,..,l]}$ of the members of the group. For example, suppose $l = 4$ and A_1 wants to sign a contract, say the message M, with A_2 and A_4. A possible message sequence is:

1. $A_1 \rightarrow A_2 : \ Exp(M, k_1)$
2. $A_2 \rightarrow A_4 : \ Exp(Exp(M, k_1), k_2)$
3. $A_4 \rightarrow A_1 : \ Exp(Exp(Exp(M, k_1), k_2), k_4)$

On receiving the second message, A_4 can verify the signatures and identity of the agents that have signed M by testing whether

$$Exp(Exp(Exp(Exp(M, k_1), k_2), K_1), K_2) = Exp(M, k_1 \cdot K_1 \cdot k_2 \cdot K_2) = M.$$

Agent A_4 can then also sign the contract using her private key k_4. The point here is that due to the commutativity property, A_4 does not need to know in what order the agents signed the message. Certainly, this protocol, when for instance used as a contract signing protocol, has many problems, which, however, we do not intend to discuss here.

7.2 The Protocol and Intruder Model for Protocols with Commutative Public-Key Encryption

We now provide a formal definition of our model by defining terms, messages, protocols, the intruder, and attacks.

Terms and Messages.. The definitions are similar to the ones in Section 2.1. We omit the public-key operator $\{m\}^p_k$ as it is now represented by $Exp(m, k)$. We could include it to model non-commutative encryption. However, for brevity of presentation, we will drop this operator. The main difference is that the product exponents are now restricted to be non-negative integers. This is motivated by the fact that, unlike in the Diffie-Hellman setting, inverting exponents is infeasible (see below for more explanation). Formally, we define:

$$\begin{aligned} term &::= \mathcal{A} \mid \mathcal{V} \mid \langle term, term \rangle \mid \{term\}^s_{term} \mid Exp(term, product) \\ product &::= term^{\mathbb{N}} \mid term^{\mathbb{N}} \cdot product \end{aligned}$$

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 $\mathbb{N}$ is the set of non-negative integers. We assume that there is a bijection $\cdot'$ on $\mathcal{K}$ which maps every public (private) key k to its corresponding private (public) key k'.

As mentioned, the exponentiation operator now models commutative public-key encryption. Therefore, the product exponents are restricted to be non-negative integers since it is infeasible to decrypt a message $Exp(m, k)$ without knowing the private key k' of k even if one has the public-key k. (Recall that in the case of Diffie-Hellman Exponentiation, given k, everyone, including the intruder, could compute k^{-1} and then $Exp(Exp(m, k), k^{-1}) = Exp(m, k \cdot k^{-1}) = Exp(m, 1) = m$.) However,

in case a principal knows the private key k', he can invert the exponent. To capture this, we consider private keys as atomic messages k' (rather than the inverse k^{-1} of k) and extend the normalization function to make sure that in exponents public and private keys cancel each other out, i.e., we have that $Exp(Exp(m,k),k') = Exp(m,k\cdot k') = Exp(m,1) = m$.

More precisely, we consider the following algebraic properties, which include the ones for Diffie-Hellman exponentiation (Section 2.1) and in addition the identity $k\cdot k' = 1$ where k' is the private (public) key corresponding to the public (private) k. Thus, besides commutativity and associativity of the product operator we consider the following properties where t is a standard term, M_1, M_2 are products, $k, k' \in \mathcal{K}$ as above, and z, z' are *non-negative* integers:

$$\begin{array}{llll} t^1 = t & t\cdot 1 = t & & Exp(t,1) = t \\ t^0 = 1 & t^z\cdot t^{z'} = t^{z+z'} & & Exp(Exp(t,M_1),M_2) = Exp(t,M_1\cdot M_2) \\ 1^z = 1 & k\cdot k' = 1 & & \end{array}$$

A *normal form* $\lceil t \rceil$ of a term t is defined analogously to the case of Diffie-Hellman exponentiation, i.e., it is obtained by exhaustively applying these identities from left to right. Note that $\lceil t \rceil$ is uniquely determined up to commutativity and associativity of the product operator. Two terms t and t' are *equivalent* if $\lceil t \rceil = \lceil t' \rceil$. The notion of normal form extends in the obvious way to sets of terms and substitutions. We illustrate the notion of a normal form by some examples: If $a, b, c, d \in \mathcal{K}$, then

(1) $\lceil (a^2\cdot b^1)\cdot b'^2 \rceil = a^2\cdot b'$,

(2) $\lceil Exp(Exp(a,(b^1\cdot c^1), c'\cdot d'^2) \rceil = Exp(a, b\cdot d'^2)$, and

(3) $\lceil Exp(Exp(a, b^3\cdot c'^6\cdot b'^3), c^6) \rceil = a$.

Recall that, for instance, b' denotes the decryption key corresponding to b.

Protocols. Protocols are defined just as in Definition 2.6.

In our protocol model, the RSA protocol (Section 7.1) can formally be stated as follows where we assume that A runs one instance of the protocol as initiator and B runs one instance as responder. The protocol consists of three protocol rules denoted $(A,1)$, $(A,2)$, and $(B,1)$ with

$$\begin{array}{lll} (A,1) & : & 1 \Rightarrow Exp(\mathsf{secret}, K_A), \\ (A,2) & : & x \Rightarrow Exp(x, K'_A), \quad \text{and} \\ (B,1) & : & y \Rightarrow Exp(y, K_B) \end{array}$$

where $(A,1)$ and $(A,2)$ denote the first and second protocol step performed by A, respectively, and $(B,1)$ denotes B's protocol step. The partial ordering is $<= \{((A,1),(A,2))\}$, i.e., we only have that $(A,1) < (A,2)$. This makes sure that $(A,1)$ must be performed before $(A,2)$. The initial intruder knowledge is $\{1, K_I, K'_I\}$, i.e., besides the constant 1, the intruder knows his public and private key.

The Intruder Model and Attacks.. Given a finite normalized set E of messages, the (infinite) set $forge(E)$ of messages the intruder can derive from E is defined in the same way as it is defined for the case of Diffie-Hellman Exponentiation except

that the product exponents z_i in the oracle rules are now restricted to be *non-negative* integers (cf. Definition 4.1). The intruder obtained in this way is called the *RSA intruder* in what follows.

Attacks and the problem INSECURE are defined as before (see Definition 2.8).

It can easily be checked that the protocol formally specified above is insecure according to our definition.

7.3 Main Results for Protocols with Commutative Public-Key Encryption

The following results carry over from the case of Diffie-Hellman Exponentiation in a rather straightforward way.

THEOREM 7.1. *For the RSA intruder,* DERIVE *can be decided in deterministic polynomial time.*

The proof of this theorem is along the same lines as the one for the DH intruder.

THEOREM 7.2. *For the RSA intruder, the problem* INSECURE *is NP-complete.*

The main difference to the proof for the DH intruder is that now we do not reduce the insecurity problem to solving linear equations in integers, but non-negative integers. Since, according to [Borsh and Treybig 1976], the size of the solutions can still be bounded polynomially in the size of the equation system, we still can bound the size of the substitution needed for an attack, and hence, obtain an NP decision algorithm. As before, NP-hardness is easily established.

8. CONCLUSION

We have shown that the insecurity problem for protocols that use Diffie-Hellman exponentiation with arbitrary products in exponents is NP-complete and that in this setting the derivation problem can be decided in deterministic polynomial time. We have also shown how these results can be transfered to protocols with commuting public key encryption.

REFERENCES

AMADIO, R., LUGIEZ, D., AND VANACKERE, V. 2002. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science 290,* 1, 695–740.

BASIN, D., MÖDERSHEIM, S., AND VIGANÒ, L. 2003. An On-The-Fly Model-Checker for Security Protocol Analysis. In *Proceedings of the 8th European Symposium on Research in Computer Security (ESORICS 2003)*, E. Snekkenes and D. Gollmann, Eds. Lecture Notes in Computer Science, vol. 2808. Springer, 253–270.

BOCKMAYR, A. AND WEISPFENNING, V. 2001. Solving numerical constraints. In *Handbook of Automated Reasoning*, A. Robinson and A. Voronkov, Eds. Vol. I. Elsevier Science, Chapter 12, 751–842.

BOREALE, M. 2001. Symbolic trace analysis of cryptographic protocols. In *Automata, Languages and Programming, 28th International Colloquium (ICALP 2001)*. Lecture Notes in Computer Science, vol. 2076. Springer-Verlag, 667–681.

BOREALE, M. AND BUSCEMI, M. 2003. On the Symbolic Analysis of Low-Level Cryptographic Primitives: Modular Exponentiation and the Diffie-Hellman Protocol. In *In Proceedings of the Workshop on Foundations of Computer Security (FCS 2003)*.

BORSH, I. AND TREYBIG, L. 1976. Bounds on positive integral solutions of linear diophantine equations. *Proc. Amer. Math. Soc. 55*, 299–304.

BOYD, C. AND MATHURIA, A. 2003. *Protocols for Authentication and Key Establishment*. Springer.

Bull, J. and Otway, D. 1997. The authentication protocol. Technical Report DRA/CIS3/PROJ/CORBA/SC/1/CSM/436-04/03, Defence Research Agency, Malvern, UK.

Chevalier, Y., Küsters, R., Rusinowitch, M., and Turuani, M. 2003a. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*. IEEE, Computer Society Press, 261–270.

Chevalier, Y., Küsters, R., Rusinowitch, M., and Turuani, M. 2003b. Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents. In *FSTTCS 2003: Foundations of Software Technology and Theoretical Computer Science*, P. Pandya and J. Radhakrishnan, Eds. Lecture Notes in Computer Science, vol. 2914. Springer, 124–135.

Chevalier, Y., Küsters, R., Rusinowitch, M., and Turuani, M. 2004. Deciding the Security of Protocols with Commuting Public Key Encryption. In *IJCAR 2004 Workshop W6 ARSPA Automated Reasoning for Security Protocol Analysis*.

Chevalier, Y. and Vigneron, L. 2001. A Tool for Lazy Verification of Security Protocols. In *Proceedings of the 16th IEEE Conference on Automated Software Engineering (ASE 2001)*. IEEE CS Press, 373–376.

Clark, J. and Jacob, J. 1997. *A Survey of Authentication Protocol Literature*. Web Draft Version 1.0 available from http://citeseer.nj.nec.com/.

Comon-Lundh, H. and Shmatikov, V. 2003. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*. IEEE, Computer Society Press, 271–280.

Corin, R. and Etalle, S. 2002. An Improved Constraint-Based System for the Verification of Security Protocols. In *Proceedings of the 9th International Symposium on Static Analysis (SAS 2002)*, M. Hermenegildo and G. Puebla, Eds. Lecture Notes in Computer Science, vol. 2477. Springer, 326–341.

Dolev, D. and Yao, A. 1983. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory 29,* 2, 198–208.

Goubault-Larrecq, J., Roger, M., and Verma, K. 2005. Abstraction and resolution modulo AC: How to verify Diffie-Hellman-like protocols automatically. *Journal of Logic and Algebraic Programming*. To appear.

Kapur, D., Narendran, P., and Wang, L. 2003. Analyzing protocols that use modular exponentiation: Semantic unification techniques. In *Proceedings of the 14th International Conference on Rewriting Techniques and Applications (RTA 2003)*, R. Nieuwenhuis, Ed. Lecture Notes in Computer Science, vol. 2706. Springer, 165–179.

Meadows, C. 2000. Open issues in formal methods for cryptographic protocol analysis. In *Proceedings of DISCEX 2000*. IEEE Computer Society Press, 237–250.

Meadows, C. and Narendran, P. 2002. A Unification Algorithm for the Group Diffie-Hellman Protocol. In *Workshop on Issues in the Theory of Security (WITS 2002)*.

Millen, J. and Shmatikov, V. 2003. Symbolic Protocol Analysis with Products and Diffie-Hellman Exponentiation. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop (CSFW 16)*. IEEE Computer Society, 47–61.

Millen, J. K. and Shmatikov, V. 2001. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the 8th ACM conference on Computer and Communications Security*. ACM Press, 166–175.

Paulson, L. 1997. Mechanized Proofs for a Recursive Authentication Protocol. In *10th IEEE Computer Security Foundations Workshop (CSFW-10)*. IEEE Computer Society Press, 84–95.

Pereira, O. and Quisquater, J.-J. 2001. A Security Analysis of the Cliques Protocols Suites. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW-14)*. 73–81.

Pereira, O. and Quisquater, J.-J. 2004. Generic Insecurity of Cliques-Type Authenticated Group Key Agreement Protocols. In *Proceedings of the 17-th IEEE Computer Security Foundations Workshop (CSFW-17 2004)*. IEEE Computer Society Press, 16–29.

Rusinowitch, M. and Turuani, M. 2001. Protocol Insecurity with Finite Number of Sessions is NP-complete. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*. IEEE Computer Society, 174–190.

Ryan, P. and Schneider, S. 1998. An Attack on a Recursive Authentication Protocol. *Information Processing Letters 65,* 1, 7–10.

Schneier, B. 1996. *Applied Cryptography*. John Wiley& sons, New York.

Shmatikov, V. 2004. Decidable Analysis of Cryptographic Protocols with Products and Modular Exponentiation. In *13th European Symposium on Programming (ESOP 2004)*, D. Schmidt, Ed. Lecture Notes in Computer Science, vol. 2986. Springer, 355–369.

Steiner, M., Tsudik, G., and Waidner, M. 1998. CLIQUES: A new approach to key agreement. In *IEEE International Conference on Distributed Computing Systems*. IEEE Computer Society Press, 380–387.

9. APPENDIX

9.1 Characterizing the Factors of Minimal Attacks

Lemma 3.4. *Let u be a normalized term, M, M' be two products such that for all $t \in \mathcal{F}(M)$ ($t \in \mathcal{F}(M')$), t is normalized. Let s be a standard normalized term and δ the replacement $[s \leftarrow 1]$. Then:*

(1) $\ulcorner(M \cdot M')\delta\urcorner = \ulcorner\urcorner M \cdot M^{\frown}\delta\urcorner$, *in particular,* $\ulcorner M\delta\urcorner = \ulcorner\urcorner M\urcorner\delta\urcorner$.

(2) $\ulcorner Exp(u, M)\delta\urcorner = \ulcorner\urcorner Exp(u, M)\urcorner\delta\urcorner$ *if* $s \neq \ulcorner Exp(u, M)\urcorner$ *and, in case s is of the form $Exp(\cdot, \cdot)$, also $s \neq u$.*

Proof. Statement 1 is straightforward. We prove 2. given the restrictions on s.

First, assume that u is *not* of the form $Exp(\cdot, \cdot)$. Then i) $\ulcorner Exp(u, M)\urcorner = u$, and thus, $\ulcorner M\urcorner = 1$, or ii) $\ulcorner Exp(u, M)\urcorner = Exp(u, \ulcorner M\urcorner)$ and $\ulcorner M\urcorner \neq 1$. We consider both cases.

In case i) we obtain that $\ulcorner M\urcorner\delta = 1$. By 1. we know that $\ulcorner M\delta\urcorner = \ulcorner\urcorner M\urcorner\delta\urcorner (= 1)$. Thus,

$$\begin{aligned} \ulcorner Exp(u, M)\delta\urcorner &= \ulcorner Exp(u\delta, M\delta)\urcorner \quad (*) \\ &= \ulcorner Exp(u\delta, \ulcorner M\delta\urcorner)\urcorner \\ &= \ulcorner u\delta\urcorner \\ &= \ulcorner\urcorner Exp(u, M)\urcorner\delta\urcorner \end{aligned}$$

where in (*) we use that $Exp(u, M) \neq s$ (otherwise $\ulcorner Exp(u, M)\urcorner = s$ since s is normalized).

In case ii), we obtain

$$\begin{aligned} \ulcorner\urcorner Exp(u, M)\urcorner\delta\urcorner &= \ulcorner Exp(u, \ulcorner M\urcorner)\delta\urcorner \\ &= \ulcorner Exp(u\delta, \ulcorner M\urcorner\delta)\urcorner \qquad (*) \\ &= \ulcorner Exp(u\delta, \ulcorner\urcorner M\urcorner\delta\urcorner)\urcorner \\ &= \ulcorner Exp(u\delta, \ulcorner M\delta\urcorner)\urcorner \qquad (**) \\ &= \ulcorner Exp(u\delta, M\delta)\urcorner \\ &= \ulcorner Exp(u, M)\delta\urcorner \qquad (***) \end{aligned}$$

where in (*) we use that $Exp(u, \ulcorner M\urcorner) \neq s$ (otherwise $\ulcorner Exp(u, M)\urcorner = \ulcorner Exp(u, \ulcorner M\urcorner)\urcorner = s$), in (**) we use 1., and in (***) that $Exp(u, M) \neq s$. Note that both in i) and ii) the fact that $u \neq s$ in case s is of the form $Exp(\cdot, \cdot)$ is not needed.

Now, assume that $u = Exp(v, M')$ for some v and M'. Then, we obtain

$$\begin{aligned}
\ulcorner\urcorner Exp(u, M)\urcorner\delta\urcorner &= \ulcorner\urcorner Exp(v, M' \cdot M)\urcorner\delta\urcorner \\
&= \ulcorner Exp(v, M' \cdot M)\delta\urcorner && (*) \\
&= \ulcorner Exp(v\delta, (M'\delta \cdot M\delta)\urcorner && (**) \\
&= \ulcorner Exp(Exp(v\delta, M'\delta), M\delta)\urcorner \\
&= \ulcorner Exp(u\delta, M\delta)\urcorner && (***) \\
&= \ulcorner Exp(u, M)\delta\urcorner && (****)
\end{aligned}$$

where (*) is obtained just as in the first case. We use that v is not of the form $Exp(\cdot, \cdot)$ and that $Exp(v, M' \cdot M) \neq s$ (otherwise $\ulcorner Exp(v, M' \cdot M)\urcorner = \ulcorner Exp(u, M)\urcorner = s$). Recall that the first case works even without the assumption that $v \neq s$. In (**), again we use that $Exp(v, M' \cdot M) \neq s$. In (***), we use that $u \neq s$ (not that otherwise $u = s$ and s is of the form $Exp(\cdot, \cdot)$). Finally, (****) uses that $Exp(u, M) \neq s$. □

LEMMA 3.5. *Let σ be a normalized ground substitution, E a set of normalized terms, s a normalized standard non atomic term, and δ the replacement $[s \leftarrow 1]$. Let $\sigma' = \ulcorner\sigma\delta\urcorner$. If there is no standard subterm t of E such that $t \sqsubseteq_\sigma s$, then $\ulcorner E\sigma'\urcorner = \ulcorner\urcorner E\sigma\urcorner\delta\urcorner$.*

PROOF. Assume there is no standard subterm t of E such that $t \sqsubseteq_\sigma s$. Consider $\Omega_s = \{t \in \mathcal{S}(E) \,|\, \ulcorner t\sigma'\urcorner \neq \ulcorner\urcorner t\sigma\urcorner\delta\urcorner\}$. By contradiction assume the stronger result $\Omega_s \neq \emptyset$. Let $u \in \Omega_s$ be minimal for the subterm relation. By definition of σ' the term u is not a variable, and s non-atomic implies it cannot be a constant.

If $u = \{u_1\}^s_{u_2}, \{u_1\}^a_{u_2}$ or $\langle u_1, u_2\rangle$ then $\ulcorner u\sigma\urcorner \neq s$ for otherwise $u \sqsubseteq_\sigma s$. By minimality of u one has $\ulcorner u_1\sigma'\urcorner = \ulcorner\urcorner u_1\sigma\urcorner\delta\urcorner$ and $\ulcorner u_2\sigma'\urcorner = \ulcorner\urcorner u_2\sigma\urcorner\delta\urcorner$. Thus we have $\ulcorner u\sigma'\urcorner = \ulcorner\urcorner u\sigma\urcorner\delta\urcorner$ which contradicts $u \in \Omega_s$.

Thus necessarily one has $u = Exp(u_1, M)$. By minimality for all $v \in \mathcal{F}(u)$ one has $\ulcorner v\sigma'\urcorner = \ulcorner\urcorner v\sigma\urcorner\delta\urcorner$. Thus we have:

$$Exp(\ulcorner u_1\sigma'\urcorner, \ulcorner M\sigma'\urcorner) = Exp(\ulcorner\urcorner u\sigma\urcorner\delta\urcorner, \ulcorner\urcorner M\sigma\urcorner\delta\urcorner)$$

Since we have $\ulcorner u\sigma\urcorner \neq s$ and $\ulcorner u_1\sigma\urcorner \neq s$ we conclude by Lemma 3.4 point 2. □

LEMMA 3.6. *Let $t', t_1, \ldots, t_n, t, u$ be normalized standard terms, $z_1 \ldots, z_n \in \mathbb{Z}$, and let δ be the replacement $[u \leftarrow 1]$ such that $u \neq t$, and $t = \ulcorner Exp(t', t_1^{z_1} \cdots t_n^{z_n})\urcorner$. If $t' = Exp(\cdot, \cdot)$, then we also assume that $u \neq t'$. Then,*

$$\ulcorner t\delta\urcorner = \ulcorner Exp(\ulcorner t'\delta\urcorner, \ulcorner t_1\delta\urcorner^{z_1} \cdots \ulcorner t_n\delta\urcorner^{z_n})\urcorner.$$

PROOF. We distinguish two cases. First, assume that $t' = Exp(v, M)$ for some normalized term v and a normalized product M. Note that $v \neq Exp(\cdot, \cdot)$ since t' is normalized.Then,

$$\begin{aligned}
\ulcorner Exp(\ulcorner t'\delta\urcorner, \ulcorner t_1\delta\urcorner^{z_1} \cdots \ulcorner t_n\delta\urcorner^{z_n})\urcorner &= \ulcorner Exp(\ulcorner v\delta\urcorner, \ulcorner M\delta\urcorner \cdot \ulcorner t_1\delta\urcorner^{z_1} \cdots \ulcorner t_n\delta\urcorner^{z_n})\urcorner && (*) \\
&= \ulcorner Exp(v\delta, M\delta \cdot (t_1\delta)^{z_1} \cdots (t_n\delta)^{z_n})\urcorner \\
&= \ulcorner Exp(v, M \cdot t_1^{z_1} \cdots t_n^{z_n})\delta\urcorner && (**) \\
&= \ulcorner\urcorner Exp(v, M \cdot t_1^{z_1} \cdots t_n^{z_n})\urcorner\delta\urcorner && (***) \\
&= \ulcorner t\delta\urcorner
\end{aligned}$$

where in (*) we use that $u \neq t'$, and thus, $\ulcorner t'\delta \urcorner = \ulcorner v\delta \urcorner$ (in this case, $\ulcorner M\delta \urcorner = 1$) or $\ulcorner t'\delta \urcorner = Exp(\ulcorner v\delta \urcorner, \ulcorner M\delta \urcorner)$. In (**) we use that $u \neq t$: If $u = Exp(v, M{\cdot}t_1^{z_1} \cdots t_n^{z_n})$, then since u is normalized $Exp(v, M{\cdot}t_1^{z_1} \cdots t_n^{z_n})$ must be normalized, but then we have that $u = Exp(v, M{\cdot}t_1^{z_1} \cdots t_n^{z_n}) = \ulcorner Exp(v, M{\cdot}t_1^{z_1} \cdots t_n^{z_n}) \urcorner = t$, in contradiction to $u \neq t$. Finally, in (***) we use that $v \neq Exp(\cdot, \cdot)$, $u \neq t$, and Lemma 3.4, 2.

Now, assume that $t' \neq Exp(\cdot, \cdot)$. For both cases, $u \neq t'$ and $u = t'$, the argument is similar to the one above. (Replace v by t' and omit $\ulcorner M\delta \urcorner$, $M\delta$, and M in the above identities.) □

LEMMA 3.12. *Let $t \in forge(E)$ and $\gamma \in forge(E)$ be given with a derivation D_γ from E ending with an application of a rule in $\mathcal{L}_c$. Then, there is a derivation D' from E with goal t satisfying $L_d(\gamma) \notin D'$.*

PROOF. First, we need some notation. If $D_1 = E_1 \to \ldots \to F_1$ and $D_2 = E_2 \to \ldots \to F_2$ are two derivations such that $E_2 \subseteq F_1$, then $D = D_1.D_2$ is defined as the *concatenation* of the steps of D_1 and the ones in D_2. In addition, to obtain a derivation, we remove in D the steps from D_2 that generate terms already present in F_1.

By definition of a derivation, $L_d(\gamma) \notin D_\gamma$. Let D be D_γ without its last rule, i.e., D_γ is D followed by some $L \in \mathcal{L}_c$. Define $D'' = D.Deriv_t(E) = D.D'''$ — D''' is obtained from $Deriv_t(E)$ by removing redundant steps. Note that D'' is a derivation with goal t. We distinguish two cases:

—Assume $L = L_c(\gamma)$. Then $L_d(\gamma) \notin D''$ since the (two) direct subterms of γ are created in D, and thus, $L_d(\gamma) \notin D''$. In other words, $D' = D''$ is the derivation we are looking for.

—Assume $L = L_{oc}(\gamma)$. Then, if $L_d(\gamma) \notin D''$, nothing is to show. Otherwise, let F_1 be the final set of messages of D. Now, Definition 2.11, (2) implies that every step in D''' of the form $F_1, F_2, \gamma \to_{L_d(\gamma)} F_1, F_2, \gamma, \beta$ can be replaced by a derivation from F_1, F_2 with goal β that does not contain rules from $L_d(\gamma)$. Replacing steps in this way and then removing redundant steps yields the derivation D' we are looking for. □

9.2 Extending the Dolev-Yao Intruder by Diffie-Hellman Exponentiation

LEMMA 4.3. *Let E be a finite set of normalized standard messages and t be a standard message such that t can be derived from t (w.r.t. $\mathcal{L}$). Let D be a derivation from E with goal t. Then, there exists a derivation D' from E with goal t such that*

(1) *D' is of the same length as D, and*
(2) *for every DH rule $L \in D' \cap L_o$ with head t' we have that $t' \in E$ or there exists a t'-rule $L' \in D' \cap (L_d \cup L_c)$. Moreover, if L is a decomposition DH rule, then $t' \in E$ or there exists a t'-rule $L' \in D' \cap L_d$.*

PROOF. Let D be a derivation from E with goal t. From D we construct D' as follows. Assume that $L \in D \cap L_o$ with head t' and that neither $t' \in E$ nor there exists a t'-rule $L' \in D' \cap (L_d \cup L_c)$. Then, there exists $L' \in D \cap L_o(t')$. Assume that L is of the form $t', t_1, \ldots, t_n \to t$ with product exponents $z_1, \ldots, z_n$ and that L' is of the form $t'', t_1', \ldots, t_{n'}' \to t'$ with product exponents $z_1', \ldots, z_{n'}'$.

Then, obviously $t = \ulcorner Exp(t'', t_1^{z_1} \cdots t_n^{z_n} \cdot {t'}_1^{z'_1} \cdots {t'}_{n'}^{z'_{n'}}) \urcorner$ and t can be obtained by the DH rule $\hat{L} = t'', t_1, \ldots, t_n, t'_1, \ldots, t'_{n'} \to t$ with head t''. Thus, L can be replaced by $\hat{L}$. Iterating this replacement we obtain D' satisfying 1. and for every DH rule $L \in D' \cap L_o$ with head t' there exists no $L' \in D' \cap L_o(t')$ preceding L in D'. From this 2. immediately follows. Note that if L is a decomposition DH rule, then t' is of the form $Exp(\cdot, \cdot)$, and thus, cannot be created by a rule in L_c. □

LEMMA 4.4. *Let $D = E_0 \to_{L_1} \ldots E_{n-1} \to_{L_n} E_n$ be a derivation with goal g.*

(1) *Assume that for every j with $E_{j-1} \to_{L_j} E_{j-1}, t$ the jth step in D and $L_j \in \mathcal{L}_d(t)$, there exists $t' \in E_{j-1}$ such that t is a subterm of t' and either $t' \in E_0$ or there exists i with $i < j$ and $L_i \in \mathcal{L}_d(t')$. Then, if $L \in D \cap \mathcal{L}_d(t)$ for some L and t, then $t \in \mathcal{S}(E_0)$.*

(2) *Assume that for every $i < n$ and t with $L_i \in \mathcal{L}_c(t)$, there exists j with $i < j$ such that L_j is a t'-rule and $t \in \mathcal{S}(\{t'\} \cup E_0)$. Then, if $L \in D \cap \mathcal{L}_c(t)$ for some L and t, then $t \in \mathcal{S}(E_0, g)$.*

Given both the assumptions in 1. and 2., it follows that D is a well-formed derivation with goal g.

PROOF. 1. is immediate by induction on $j \in \{1, \ldots, n\}$. Given the assumptions in 2., we prove by induction on $n - i$ that for all $i \in \{1, \ldots, n\}$, $L_i \in \mathcal{L}_c(t)$ implies $t \in \mathcal{S}(E_0, g)$. If $n - i = 0$, then $t = g$ and therefore $t \in \mathcal{S}(E_0, g)$. For the induction step, the assumptions in 2. imply that there exists $j > i$ such that L_j is a t'-rule and $t \in \mathcal{S}(E_0, t')$. If $L_j \in \mathcal{L}_d(t')$, then $t' \in \mathcal{S}(E_0)$ (see above). If $L_j \in \mathcal{L}_c(t')$, then by induction $t' \in \mathcal{S}(E_0, g)$, and hence, $t \in \mathcal{S}(E_0, g)$.

Given both the assumptions in 1. and 2., it immediately follows that D is a well-formed derivation with goal g. □

LEMMA 4.6. *Let $z_1, \ldots, z_n \in \mathbb{Z} \setminus \{0\}$, and $s, s_1, \ldots, s_n, u$ be normalized standard terms such that $s_i \neq s_j$ for every $i \neq j$, $s_i \neq 1$ and $s_i \neq u$ for every i, $s \neq u$, $u = \ulcorner Exp(s, s_1^{z_1} \cdots s_n^{z_n}) \urcorner$, and $u = Exp(\cdot, \cdot)$. Let δ be the replacement $[u \to 1]$. Then, $u = \ulcorner Exp(\ulcorner s\delta \urcorner, \ulcorner s_1\delta \urcorner^{z_1} \cdots \ulcorner s_n\delta \urcorner^{z_n}) \urcorner$.*

PROOF. First, assume that $s \neq Exp(\cdot, \cdot)$. Then, $u = Exp(s, s_1^{z_1} \cdots s_n^{z_n})$. Consequently, $u \notin \mathcal{S}(s, s_1, \ldots, s_n)$, and thus, $s = s\delta$, and $s_i = s_i\delta$ for every i. Therefore, we obtain that $u = \ulcorner Exp(\ulcorner s\delta \urcorner, \ulcorner s_1\delta \urcorner^{z_1} \cdots \ulcorner s_n\delta \urcorner^{z_n}) \urcorner$.

Now, assume that $s = Exp(v, M)$. Since s is normalized we know that $v \neq Exp(\cdot, \cdot)$. Using that $u = Exp(\cdot, \cdot)$, we obtain $u = Exp(v, \ulcorner M \cdot s_1^{z_1} \cdots s_n^{z_n} \urcorner)$ with $\ulcorner M \cdot s_1^{z_1} \cdots s_n^{z_n} \urcorner \neq 1$. Also, $u \notin \mathcal{S}(v)$. Then, with $E = \mathcal{F}(M) \cup \{s_1, \ldots, s_n\}$ there exists a set $E' = \{s'_1, \ldots, s'_{n'}\} \subseteq E$ and $z'_1, \ldots, z'_{n'} \in \mathbb{Z} \setminus \{0\}$ such that $u = Exp(v, {s'}_1^{z'_1} \cdots {s'}_{n'}^{z'_{n'}})$ and $u \notin \mathcal{S}(v, E')$.

Claim. $\ulcorner \ulcorner M\delta \urcorner \cdot \ulcorner s_1\delta \urcorner^{z_1} \cdots \ulcorner s_n\delta \urcorner^{z_n} \urcorner = {s'}_1^{z'_1} \cdots {s'}_{n'}^{z'_{n'}}$.

Proof of the claim. Assume that $M = s_{n+1}^{z_{n+1}} \cdots s_{n''}^{z_{n''}}$. Let:

$$C_i = \{j \in \{1, \ldots, n''\} \mid s_j = s'_i\}.$$

Then, $z'_i = \Sigma_{j \in C_i} z_j$. Let $C = \bigcup_{i=1}^{n'} C_i$. We have that $\ulcorner s_1^{z_1} \cdots s_n^{z_n} \cdot s_{n+1}^{z_{n+1}} \cdots s_{n''}^{z_{n''}} \urcorner =$

$\ulcorner \Pi_{i=1}^{n'} \Pi_{j \in C_i} s_j^{z_j \urcorner} = \Pi_{i=1}^{n'} {s'}_i^{z'_i}$ and $\ulcorner \Pi_{j \notin C} s_j^{z_j \urcorner} = 1$. Using that the s_j are normalized, it follows that $\ulcorner \Pi_{j \notin C} s_j \delta^{z_j \urcorner} = \ulcorner \Pi_{j \notin C} {}^{\ulcorner} s_j \delta^{\urcorner z_j \urcorner} = 1$. Now with $s'_i \delta = s'_i$ the claim follows.

Using that $s\delta = Exp(v\delta, M\delta)$, $u \neq M$, $u \notin \mathcal{S}(v)$, and $u \neq s$, the claim implies that

$$\ulcorner Exp(\ulcorner s\delta \urcorner, \ulcorner s_1 \delta \urcorner^{z_1} \cdots \ulcorner s_n \delta \urcorner^{z_n}) \urcorner = \ulcorner Exp(\ulcorner v\delta \urcorner, \ulcorner M\delta \urcorner \cdot \ulcorner s_1 \delta \urcorner^{z_1} \cdots \ulcorner s_n \delta \urcorner^{z_n}) \urcorner = u.$$

□

9.3 The DH Rules Allow Polynomial Product Exponent Attacks

LEMMA 5.11. *For every open message or product t such that $\beta(t) = \ulcorner \beta(t) \urcorner$, we have that $t^\beta = t$.*

PROOF. The proof proceeds by structural induction on t. If t is atomic, a pair, or encryption, the claim is obvious.

If $t = t_1^{e_1} \cdots t_n^{e_n}$, we know that that $\beta(t_i) = \ulcorner \beta(t_i) \urcorner \neq 1$ for every i, $\ulcorner \beta(t_i) \urcorner \neq \ulcorner \beta(t_j) \urcorner$ for every $i \neq j$, and $\beta(e_i) \neq 0$ for every i. From this, one easily concludes that $t^\beta = t$.

Finally, assume that $t = Exp(u, M)$. First we note that $\ulcorner \beta(u) \urcorner \neq Exp(u', M')$ for any normalized u' and M': Otherwise, $\ulcorner \beta(t) \urcorner = Exp(u', M'') = \beta(t)$, and thus, $\beta(u) = u'$. Hence, $Exp(u', M') = \ulcorner \beta(u) \urcorner = \ulcorner u' \urcorner = u'$, which is a contradiction. Also, $\ulcorner \beta(M) \urcorner \neq 1$ since otherwise $Exp(\beta(u), \beta(M)) = \beta(t) = \ulcorner \beta(t) \urcorner = \ulcorner \beta(u) \urcorner$, but we know that $\ulcorner \beta(u) \urcorner \neq Exp(\cdot, \cdot)$. Hence, $Exp(\beta(u), \beta(M)) = \beta(t) = \ulcorner \beta(t) \urcorner = Exp(\ulcorner \beta(u) \urcorner, \ulcorner \beta(M) \urcorner)$, and thus, $\beta(u) = \ulcorner \beta(u) \urcorner$ and $\beta(M) = \ulcorner \beta(M) \urcorner$. By induction, this yields $u^\beta = u$ and $M^\beta = M$. By definition of t^β, we obtain $t^\beta = Exp(u^\beta, M^\beta) = Exp(u, M) = t$. □

LEMMA 5.13. *Let E be a finite set of open messages or products such that $\mathcal{S}_{ext}(E) = E$ and t maximal (w.r.t. subterm ordering) in E. Then:*

$$|\bigcup_{s \in E} \mathcal{S}_{ext}(s^\beta)|_{exp} \leq |\bigcup_{s \in E \setminus \{t\}} \mathcal{S}_{ext}(s^\beta)|_{exp} + ||t||_{ext}^2$$

To prove this lemma, we first need to bound the $|...|_{exp}$ size of a product. This is the following claim:

CLAIM 1. *For every open message or product t with $t^\beta = Exp(u, M)$ it follows that $|M|_{exp} \leq |t| \cdot ||t||_{exp} \leq ||t||_{ext}^2$.*

PROOF. We show by structural induction on t that $|M|_{exp} \leq |t| \cdot ||t||_{exp}$. Since $|t| \leq ||t||_{ext}$ and $||t||_{exp} \leq ||t||_{ext}$, the lemma follows.

First consider the case were $t = Exp(u', M')$ and $\ulcorner \beta(u') \urcorner \neq Exp(\cdot, \cdot)$. Then, $|M|_{exp} \leq |M'|_{exp} \leq ||t||_{exp}$.

Now, assume that $t = Exp(u', M')$ and $\ulcorner \beta(u') \urcorner = Exp(\cdot, \cdot)$, and thus, $u'^\beta = Exp(u'', M'')$ for some u'' and M''. Then, $M = (M'' \cdot M')^\beta$, and therefore, by induction $|M|_{exp} \leq |M''|_{exp} + |M'|_{exp} \leq |u'| \cdot ||u'||_{exp} + ||t||_{exp} \leq |t| \cdot ||t||_{exp}$ where we use that $||u'||_{exp} \leq ||t||_{exp}$ and $|u'| < |t|$.

In all other cases, if $t^\beta = Exp(u, M)$, then there exists a subterm v of t such that $v^\beta = t^\beta$. In these cases the bound follows by induction. □

We can now proceed with the proof of Lemma 5.13 :

PROOF. We proceed (again) by structural induction on t. If t is an atomic message, then obviously $|\bigcup_{s\in E}\mathcal{S}_{ext}(s^\beta)|_{exp} = |\bigcup_{s\in E\setminus\{t\}}\mathcal{S}_{ext}(s^\beta)|_{exp}$.

If $t = \langle t_1, t_2\rangle$, then $t^\beta = \left\langle t_1^\beta, t_2^\beta\right\rangle$, and thus, $\mathcal{S}_{ext}(t^\beta) \subseteq \{t^\beta\} \cup \mathcal{S}_{ext}(t_1^\beta) \cup \mathcal{S}_{ext}(t_2^\beta)$. We know that $t_1, t_2 \in E \setminus t$ since $\mathcal{S}_{ext}(E) = E$. Since $|t^\beta|_{exp} = 0$, we obtain $|\bigcup_{s\in E}\mathcal{S}_{ext}(s^\beta)|_{exp} = |\bigcup_{s\in E\setminus\{t\}}\mathcal{S}_{ext}(s^\beta)|_{exp}$. For encryption the argument is analogously.

If $t = t_1^{e_1}\cdots t_n^{e_n}$, then $\mathcal{S}_{ext}(t^\beta) \subseteq \{t^\beta\} \cup \bigcup_i \mathcal{S}_{ext}(t_i^\beta)$. We know that $t_1, \ldots, t_n \subseteq E\setminus\{t\}$, and thus, $\mathcal{S}_{ext}(t^\beta) \subseteq \bigcup_{s\in E\setminus\{t\}}\mathcal{S}_{ext}(s^\beta)\cup\{t^\beta\}$. With $|t^\beta|_{exp} \leq |t|_{exp} \leq ||t||^2_{ext}$ the claim follows.

Now, assume that $t = \mathit{Exp}(u, M)$ and $t^\beta = u^\beta$, $t^\beta = u''$, or $t^\beta = \mathit{Exp}(u^\beta, M^\beta)$ (see the cases in Lemma 5.10). In the first two cases we have $\mathcal{S}_{ext}(t^\beta) \subseteq \mathcal{S}_{ext}(u^\beta) \cup \mathcal{S}_{ext}(M^\beta) \subseteq \bigcup_{s\in E\setminus\{t\}}\mathcal{S}_{ext}(s^\beta)$. In the latter case, we have $\mathcal{S}_{ext}(t^\beta) \subseteq \{t^\beta\} \cup \mathcal{S}_{ext}(u^\beta) \cup \mathcal{S}_{ext}(M^\beta) \subseteq \{t^\beta\} \cup \bigcup_{s\in E\setminus\{t\}}\mathcal{S}_{ext}(s^\beta)$ and $|t^\beta|_{exp} = 0$. Thus, in every case $|\bigcup_{s\in E}\mathcal{S}_{ext}(s^\beta)|_{exp} = |\bigcup_{s\in E\setminus\{t\}}\mathcal{S}_{ext}(s^\beta)|_{exp}$.

Finally, assume that $t = \mathit{Exp}(u, M)$, $\ulcorner\beta(u)\urcorner = \mathit{Exp}(u', M')$, $u^\beta = \mathit{Exp}(u'', M'')$. Then, $t^\beta = \mathit{Exp}(u'', (M''\cdot M)^\beta)$, and thus,

$$\begin{aligned}\mathcal{S}_{ext}(t^\beta) &\subseteq \{t^\beta\} \cup \mathcal{S}_{ext}(u^\beta) \cup \mathcal{S}_{ext}((M''\cdot M)^\beta)\\ &\subseteq \{t^\beta\} \cup \mathcal{S}_{ext}(u^\beta) \cup \{(M''\cdot M)^\beta\} \cup \mathcal{S}_{ext}(M^\beta).\end{aligned}$$

(Note that by Lemma 5.11, $\mathcal{S}_{ext}(M''^\beta) = \mathcal{S}_{ext}(M'')$ and that $\mathcal{S}_{ext}(M'') \subseteq \mathcal{S}_{ext}(u^\beta)$.) Consequently, $\mathcal{S}_{ext}(t^\beta) \subseteq \{t^\beta\} \cup \{(M''\cdot M)^\beta\} \cup \bigcup_{s\in E\setminus\{t\}}\mathcal{S}_{ext}(s^\beta)$. We know that $|t^\beta|_{exp} = 0$, and by Lemma 1, it follows $|(M''\cdot M)^\beta|_{exp} \leq ||t||^2_{ext}$. □

LEMMA 5.17. *Let $M = t_1^{e_1}\cdots t_n^{e_n}$ and $M' = {t'}_1^{e'_1}\cdots {t'}_{n'}^{e'_{n'}}$ such that $\beta(t'_i) = \ulcorner\beta(t'_i)\urcorner$. Let $(t_i^\beta, \mathcal{E}_i)$ be a β-tuple for t_i for every i. Then, $((M'\cdot M)^\beta, {\mathcal{E}'}^\beta_{(M'\cdot M)} \cup \bigcup_i \mathcal{E}_i)$ is a β-tuple for $M'\cdot M$.*

PROOF. Let $t = M'\cdot M$. Obviously, $\beta \models {\mathcal{E}'}^\beta_t \cup \bigcup_i \mathcal{E}_i$. We also know that $\beta(t^\beta) = \ulcorner\beta(t)\urcorner$. Let $\beta' \models {\mathcal{E}'}^\beta_t \cup \bigcup_i \mathcal{E}_i$. We need to show that $\ulcorner\beta'(t^\beta)\urcorner = \ulcorner\beta'(t)\urcorner$.

Claim I. $\ulcorner\beta'(t_i)\urcorner = \ulcorner\beta'(t_i^\beta)\urcorner$ and $\ulcorner\beta'(t'_j)\urcorner = \ulcorner\beta'({t'}_j^\beta)\urcorner$ for every i and j.

Proof of Claim I. Since $\beta' \models \mathcal{E}_i$ we immediately have $\ulcorner\beta'(t_i)\urcorner = \ulcorner\beta'(t_i^\beta)\urcorner$. Lemma 5.11 implies that ${t'}_j^\beta = t'_j$. Thus, $\ulcorner\beta'(t'_j)\urcorner = \ulcorner\beta'({t'}_j^\beta)\urcorner$.

Let the classes $C_1, \ldots, C_l$ be defined as in the proof of Lemma 5.10.

Claim II. $\ulcorner\beta'(s)\urcorner = \ulcorner\beta'(s')\urcorner$ for every k and $s, s' \in C_k$.

Proof of the Claim II. First, assume that $s = t'_i$ and $s' = t'_j$ for some i and j. Due to Lemma 5.11, we have that $s^\beta = s$ and $s'^\beta = s'$. Then, by definition of ${\mathcal{E}'}^\beta_t$ we obtain that $\beta'(s) = \beta'(s')$. Now, assume that $s = t_i$ and $s' = t'_j$. Again, we have $s'^\beta = s'$. The definition of ${\mathcal{E}'}^\beta_t$ yields that $\beta'(s') = \beta'(s^\beta)$. Since $\beta' \models \mathcal{E}_i$ we have that $\ulcorner\beta'(s)\urcorner = \ulcorner\beta'(s^\beta)\urcorner$. Thus, $\ulcorner\beta'(s')\urcorner = \ulcorner\beta'(s)\urcorner$. A similar argument can be applied if $s = t_i$ and $s' = t_j$. This concludes the proof of the claim.

By definition of ${\mathcal{E}'}^\beta_t$ we know that $\beta'(e_{C_j}) = 0$ for every $j \in J$. Using Claim I and II it is now easy to see that $\ulcorner\beta'(t^\beta)\urcorner = \ulcorner\beta'(t)\urcorner$. □

LEMMA 5.20. *Let $t_1, \ldots, t_n$ be open messages, s be a normalized message, β be an evaluation mapping, and $L \in \mathcal{L}$ an intruder rule such that $\beta(t_i) = \lceil \beta(t_i) \rceil$ for all i and $\lceil \beta(t_1) \rceil, \ldots, \lceil \beta(t_n) \rceil \to s \in L$. Then, there exists an open message t, an equation system $\mathcal{E}$, and an extension β of β such that*

(1) $\beta(t) = s$,
(2) $\beta \models \mathcal{E}$,
(3) $\lceil \beta'(t_1) \rceil \ldots, \lceil \beta'(t_n) \rceil \to \lceil \beta'(t) \rceil \in \mathcal{L}$ *for every* $\beta' \models \mathcal{E}$,
(4) $\max\{|e| \mid e \in \mathcal{L}_{exp}(t)\} \leq \max\{|e| \mid e \in \mathcal{L}_{exp}(t_1, \ldots, t_n)\} + n$, *and*
(5) *the size of* $\mathcal{E}$ *is polynomially bounded in* $||t_1, \ldots, t_n||_{ext}$.

To prove this lemma, we first prove the following claim:

Claim. Let $t_0, \ldots, t_n$ be open messages and β be an evaluating mapping such that $\beta(t_i) = \lceil \beta(t_i) \rceil$ for every i. Let $z_1, \ldots, z_n$ be integer variables not occurring in the t_i. Finally, let $t = Exp(t_0, t_1^{z_1} \cdots t_n^{z_n})^\beta$. Then, the size of every linear expression in t is bounded by $\max\{|e| \mid e \in \mathcal{L}_{exp}(t_0, t_1, \ldots, t_n)\} + n$.

Proof of the claim. First note that due to Lemma 5.11 we have $t_i^\beta = t_i$. Let $M = t_1^{z_1} \cdots t_n^{z_n}$. We distinguish different cases.

The case where $\beta(t_0) = \lceil \beta(t_0) \rceil \neq Exp(\cdot, \cdot)$ is easy to prove.

Now, assume that $\beta(t_0) = \lceil \beta(t_0) \rceil = Exp(u', M')$, and thus, $t_0 = Exp(u'', M'')$ with $\beta(u'') = u'$ and $\beta(M'') = M'$. If $t = u''$, the statement of the lemma is obvious. Otherwise, $t = Exp(u'', (M'' \cdot M)^\beta)$ and $(M'' \cdot M)^\beta \neq 1$. Assume that $M'' = {t''}_1^{e''_1} \cdots {t''}_{n''}^{e''_{n''}}$. We know that $\lceil \beta(t_i'') \rceil = \beta(t_i'')$ for every i and $\beta(t_i'') \neq \beta(t_j'')$ for every $i \neq j$. Let $C_1, \ldots, C_l$ be the equivalence classes as defined in the proof of Lemma 5.10. Then, $t = \Pi_{j \notin J \cup \{1\}} (s_{C_j}^\beta)^{e_{C_j}}$. Using that $\lceil \beta(t_i'') \rceil = \beta(t_i'')$, and thus, ${t''}_i^\beta = t_i''$, and $t_i^\beta = t_i$ we know that $s_{C_j}^\beta = t_i''$ or $s_{C_j}^\beta = t_i$ for some i. Also, there is at most one t_i'' in every class C_j. Consequently, the size of e_{C_j} is bounded as required. All proper subterms of t are subterms of some t_i. Thus, the size of linear expressions in t can be bounded as required. This concludes the proof of the claim.

We can now proceed with the proof of Lemma 5.20 :

PROOF. We consider the different intruder rules L.

First, assume that $L = L_{p1}$. Then, $n = 1$, $\lceil \beta(t_1) \rceil = \langle a, b \rangle$, $s = a$, $t_1 = \langle a', b' \rangle$ where a and b are normalized messages, and a' and b' are open messages with $\beta(a') = a$ and $\beta(b') = b$. We define $t = a'$, $\mathcal{E} = \emptyset$, and β is not extended. Then, obviously conditions 1. to 5. are satisfied. The cases for L_{p2}, L_{ad}, and L_c are similar.

Now, assume that $L = L_{sd}$. Then, $n = 2$, $\lceil \beta(t_1) \rceil = \{a\}_b^s$ and $\lceil \beta(t_2) \rceil = b$ for normalized messages a and b. Thus, $t_1 = \{a'\}_{b'}^s$ and $t_2 = b''$ where a', b', and b'' are open messages such that $\beta(a') = a$ and $\beta(b') = \beta(b'') = b$. We define $t = a'$, $\mathcal{E} = \mathcal{E}_{b',b''}^{=_\beta}$, and β is not extended. Using Lemma 5.5, it is easy to check that the conditions 1. to 5. are satisfied.

Finally, assume that $L = L_o$. Then, $s = \lceil Exp(\beta(t_1), \beta(t_2)^{a_2} \cdots \beta(t_n)^{a_n} \rceil$ for some $a_i \in \mathbb{Z}$. We define $t = Exp(t_1, t_2^{z_2} \cdots t_n^{z_n})^\beta$, where the z_i are new variables, and $\mathcal{E} = \mathcal{E}_t^\beta$. We extend β such that $\beta(z_i) = a_i$ for every i. By Lemma 5.10, we have $s = \lceil \beta(Exp(t_1, t_2^{z_2} \cdots t_n^{z_n})) \rceil = \beta(t)$, $\beta \models \mathcal{E}$, $\lceil \beta'(t) \rceil = \lceil \beta'(Exp(t_1, t_2^{z_2} \cdots t_n^{z_n})) \rceil =$

$\ulcorner Exp(\ulcorner\beta'(t_1)\urcorner, \ulcorner\beta'(t_2)\urcorner^{\beta'(z_2)} \cdots \ulcorner\beta'(t_2)\urcorner^{\beta'(z_2)})\urcorner$ for every $\beta' \models \mathcal{E}$. The claim implies 4. and from Lemma 5.18 we obtain 5. □

Received May 2005; revised October 2006; accepted February 2007