# Symbolic Analysis of Cryptographic Protocols Containing Bilinear Pairings

Alisa Pankova
Institute of Computer Science
University of Tartu, Estonia
alisa.pankova@ut.ee

Peeter Laud
Cybernetica AS
Tartu, Estonia
peeter.laud@cyber.ee

***Abstract*—Bilinear pairings are powerful mathematical structures that can be used in cryptography. Their equational properties allow constructing cryptographic primitives and protocols that would be otherwise ineffective or even impossible.**

**In formal cryptography, the protocols are expressed through term algebras and process calculi. ProVerif, one of the most successful protocol analyzers, internally converts them to Horn theories for the analysis. This approach cannot easily deal with complex equational theories.**

**In this paper, we propose an equational theory that models bilinear pairings in formal cryptography. We also propose a reduction from the derivation problem for Horn theories modulo this equational theory to (almost) purely syntactical derivation problem for Horn theories. This derivation problem can be readily tackled by ProVerif. We have implemented our analysis and have demonstrated that it is able to handle several secure and insecure protocols based on bilinear pairings.**

**Our approach mostly follows Küsters's and Truderung's handling of Diffie-Hellman exponentiation. The greater complexity of the theory for bilinear pairings introduces several complications; the arithmetic properties of exponentiation play a much bigger role in our reduction. Still, our approach has the same kind of generality as theirs. Similarly to their approach, we do not treat the group operations as (independent) term constructors. But we show that access to those operations will not increase the power of the adversary.**

## I. Introduction

The complex algebraic properties of bilinear pairings, combined with well-understood intractability assumptions make them one of the most versatile components of cryptographic primitives and protocols, having been used to construct identity- and attribute-based cryptosystems [1], [2], encryption systems with keyword search [3], designated verifier signatures [4], efficient key-agreement protocols [5], [6], etc. that would be impossible or much less efficient with other techniques. Therefore it seems surprising that there have been almost no attempts to treat pairings abstractly in the symbolic (formal, perfect) cryptography, and to subject systems employing them to formal verification.

In this work, we propose a model for bilinear pairings in symbolic cryptography. Compared to the only other model proposed so far [7], the operations it offers are much closer to what are used in computational cryptography. Our model allows for exponentiations and pairings of arbitrary values, and is sufficiently rich to enable modeling of several pairing-based protocols proposed in the literature.

We also propose a method for automatically verifying protocols containing bilinear pairings. We concentrate on verification tools based on performing derivations according to Horn theories that the protocol descriptions have been transformed into; particularly on ProVerif [8], one of the most successful tools for cryptographic protocol verification. By extending the theory transformation methods of Küsters and Truderung [9], we can change the problem of derivation *modulo* the equational theory for bilinear pairings into the problem of syntactic derivation.

Our protocol verification method inherits the strengths and weaknesses of the method of [9]. It can be used to verify basic secrecy and authentication (modeled as correspondence) properties of protocols, because these are easily expressed through Horn theories. More complex properties, expressed through observational equivalence, are not so readily expressed. Our method allows the honest participants to perform exponentiations and pairings with the values they've obtained, but not multiplications in the group (there is no such constraint for the adversary, though). Our method also requires that the number of different exponents used by honest parties is bounded. However, we have no such restriction on the number of protocol sessions the parties can execute.

This work has the following structure. After reviewing related work in Sec. II and Horn theory based protocol analysis in Sec. III, we describe our symbolic model for bilinear pairings in Sec. IV. Sec. V–VIII deal with making the inference *modulo* properties of bilinear pairings tractable for ProVerif. Sec. V extends the notion of *exponent ground* theories [9] to pairings. Sec. VI presents the ideas of theory transformation at term level, while Sec. VII and VIII show the actual transformation and prove its results sound (Sec. VIII) and complete (Sec. VII) with respect to our theory of bilinear pairings. We finish the paper with an overview of our experimental results in Sec. IX, with the analysis of the adversarial power in the presence of group operations (Sec. X) and with the conclusion in Sec. XI.

## II. Related work

We perform the analysis of protocols in the *symbolic cryptography* [10] model, using the ProVerif protocol analyzer [8]. Our reduction from Horn theories *modulo* equational theory for bilinear pairings to an almost pure Horn theory advances

the methods of Küsters and Truderung who have used similar reductions to deal with the algebraic properties of XOR [11] and Diffie-Hellman exponentiation [9] before (see these papers for a more thorough discussion on XOR- and DH-theories in protocol verification). Their reduction can handle more possible adversarial actions than the equational reasoning ProVerif itself is capable to handle (although this has also been used for certain protocol analysis [12], [13]). Recently, however, Mödersheim [14] has shown that in certain well-tagged [15] cases, ProVerif's treatment may be equivalent to Küsters and Truderung [9].

By our knowledge, bilinear pairings in symbolic cryptography have been considered only by Mazaré and Kremer [7]. They proposed a signature and an equational theory for it that did not actually contain the pairing operation $e$. Instead, common uses of pairings (by legitimate participants) in protocols were modeled by derivation rules. They showed that their extension is computationally sound against passive adversaries, thereby extending the seminal reconciliation result by Abadi and Rogaway [16].

The task considered in this paper is an instance of the general problem of inference in Horn theories *modulo* equational theories. While the static equivalence of terms is usually decidable [17], [18], it is much harder to perform an actual protocol analysis *modulo* an equational theory [13].

## III. Modeling Protocols with Horn Theories

Let $\Sigma$ be a finite signature. A *term* $t$ over $\Sigma$ is either a variable $x$ from a countable set $V$, a name $m$ from a countable set $\mathcal{N}$ or $f(t_1, \ldots, t_{\mathrm{ar}(f)})$ for $f \in \Sigma$. By $var(t)$ we denote the set of variables occurring in term $t$. A term is *ground* if it contains no variables. When modeling cryptographic protocols, the signature contains constructors corresponding to cryptographic operations. E.g. we may have a binary constructor $enc(\cdot,\cdot)$ for modeling deterministic encryption and $dec(\cdot,\cdot)$ for modeling decryption.

For a predicate $P$ of arity $k$ and terms $t_1, \ldots, t_k$, we call $P(t_1, \ldots, t_k)$ an *atom*. It is *ground* if $t_1, \ldots, t_k$ are ground. A *Horn clause* has the form $a_1, \ldots, a_n \to a_0$, where $a_0, \ldots, a_n$ are atoms. A *Horn theory* is a finite set of Horn clauses.

A ground atom $a$ can be *(syntactically) derived* in theory $T$ (denoted $T \vdash a$) if there exists a derivation $\pi = b_1, \ldots, b_l$ where $b_l = a$ and for each $j \in \{1, \ldots, l\}$, $b_j$ is a ground atom and there exists a clause $a_1, \ldots, a_n \to a_0 \in T$ and a variable substitution $\sigma$, such that $\sigma(a_0) = b_j$ and each $\sigma(a_i)$ ($i \in \{1, \ldots, n\}$) occurs in the set $\{b_1, \ldots, b_{j-1}\}$. A theory is *non-trivial* if at least one ground atom can be derived in it. If $\sim$ is a congruence relation on terms then we can speak about derivation *modulo* $\sim$ (denoted $T \vdash_\sim a$). In this case, all term equalities in the the definition of $\vdash$ are replaced with the relation $\sim$.

ProVerif [8] translates protocols into Horn theories for the purpose of verification. The main predicate symbol it uses is unary $I$, denoting intruder knowledge. The initial knowledge of the intruder is modeled as facts (clauses with zero premises) in the theory $T$. The possible operations the intruder can perform with messages are readily expressed as Horn clauses — if an attacker knows messages of certain shape, it can produce other messages from them. The messages of the protocol can also be represented as clauses — if the intruder has messages 1,...,$i$ of a protocol session, it can also obtain the message no. $i+1$ (presumably by handing some previous message over to a participant). This model introduces certain abstractions (e.g. one cannot naturally state that sessions are non-interleaving), but keeps the precision with respect to sizes of terms and number of sessions (both can be unbounded). For a protocol $P$, we let $T_P$ be the corresponding Horn theory (including intruder's initial knowledge and possible operations on messages). This theory can be used to answer certain questions about the protocol, e.g. is the intruder able to learn a certain message $m$?

## IV. Equational Theory for Bilinear Pairings

In the computational model of cryptography [19], a (symmetric) *bilinear* pairing [5], [1] is an efficiently computable non-degenerate mapping $e : G_1 \times G_1 \to G_T$, where $G_1$ (written additively) and $G_T$ (written multiplicatively) are cyclic groups of size $p$. The mapping must satisfy $e(Q_1, R_1+R_2) = e(Q_1, R_1)e(Q_1, R_2)$ and $e(Q_1+Q_2, R_1) = e(Q_1, R_1)e(Q_2, R_1)$ (*bilinearity*) for any $Q_1, Q_2, R_1, R_2 \in G_1$. As $G_1$ and $G_T$ are cyclic, we immediately obtain $e(aP, bP) = e(P, P)^{ab}$ for the generator $P$ of $G_1$ and any $a, b \in \mathbb{Z}$. Weil or Tate pairings [1], [20] are the typical examples of a bilinear pairing.

Typical intractability assumptions for cyclic groups with bilinear pairings are the computational Diffie-Hellman (CDH) assumption or the bilinear computational Diffie-Hellman (BCDH) assumption (the second implies the first). CDH assumption postulates the intractability of finding the element $abP \in G_1$ from the elements $P, aP, bP \in G_1$. BCDH assumption postulates the intractability of finding the element $e(P, P)^{abc} \in G_T$ from the elements $P, aP, bP, cP \in G_1$. In the symbolic setting, however, it makes sense to model pairing as an operation that has the bilinear property stated above, and satisfies no other equations.

For symbolically modeling bilinear pairings and exponentiations, we assume that beside other operations, the signature $\Sigma$ contains binary operations $\star$ and $\uparrow$ (for modeling multiplication in $G_1$ and exponentiation in $G_T$), unary operation $\cdot^{-1}$ (for modeling negation in $G_1$, inverse in $G_T$, and inverses in the exponents), and the binary operation $e$ (pairing). Similarly to [9] (and all other approaches so far), we omit additions in $G_1$ / multiplications in $G_T$ from our signature (see Sec. X for their treatment). Using our notation, if the term $t$ models an element $Q$ of $G_1$, and the term $u$ models an integer $n$, then the term $t \star u$ models the element $nQ$ of $G_1$. Similarly, if $t$ modeled an element $h$ of $G_T$ instead, then the term $t \uparrow u$ would model the element $h^n$ in $G_T$.

The equational theory $\sim$ for bilinear pairings is the least congruence on terms containing the following equations,

where the variables $x, y, z$ may be instantiated with any terms.

$$\begin{aligned}
(x \uparrow y) \uparrow z &\sim (x \uparrow z) \uparrow y \qquad (x \star y) \star z \sim (x \star z) \star y \\
e(x,y) &\sim e(y,x) \\
(x \uparrow y) \uparrow y^{-1} &\sim x \qquad (x^{-1})^{-1} \sim x \\
(x \star y) \star y^{-1} &\sim x \\
x^{-1} \star y &\sim (x \star y)^{-1} \qquad x^{-1} \uparrow y \sim (x \uparrow y)^{-1} \\
e(x, y \star z) &\sim e(x,y) \uparrow z \qquad e(x, y^{-1}) \sim e(x,y)^{-1}
\end{aligned} \tag{1}$$

The commutativity of $e$ actually follows from the cyclicity of $G_1$, but this notion is not available in the symbolic setting.

Associated with the operations in the signature are also the rules for intruder for composing and decomposing messages. We let the predicate $I(t)$ denote that the intruder sees the term $t$. The rules associated with bilinear pairings are

$$\begin{aligned}
I(x), I(y) &\to I(e(x,y)) \qquad I(x), I(y) \to I(x \star y) \\
I(x), I(y) &\to I(x \uparrow y) \qquad I(x) \to I(x^{-1})
\end{aligned}$$

We let $T_E$ denote the theory consisting of the rules above.

Although the groups $G_1$ and $G_T$ have different operations, the intruder has freedom to apply the operation of $G_1$ to the elements of $G_T$ and the operation of $G_T$ to the elements of $G_1$. The theory described in this work allows that, and there are no type constraints in its description. This ability, however, does not give the intruder anything useful (as we see in this paper), and therefore type constraints can be added when applying this theory with ProVerif.

Given a protocol $P$ and a message $m$, the fact that $T_P \cup T_E \nvdash_\sim I(m)$ means that the intruder cannot get the message $m$ even when employing the algebraic properties of bilinear mappings.

## V. Exponent-ground Theory for Bilinear Pairings

Küsters and Truderung [9], constrain the theory for Diffie-Hellman exponentiation $T_{DH}$ to a theory $T_{DH}^C$ that can be used only with *exponent-ground* terms. In the same way, the theory $T_E$ should be constrained to $T_E^C$. First, we need to define what does it mean for a term to be exponent-ground. We say that a term is

- *reduced*, if no equations in last four rows of (1), interpreted as reduction rules from left to right, can be applied to it *modulo* the equations in the first two rows;
- *standard*, if its head symbol is neither $\star$, $\uparrow$, $^{-1}$, nor $e$;
- *pure*, if the symbols $\star$, $\uparrow$, $^{-1}$, and $e$ do not occur in it;
- *well-formed* if each of its subterms of the form $s^{-1}$ only occurs in a context of the form $s' \uparrow s^{-1}$ or $s' \star s^{-1}$ for some $s'$;
- *exponent-ground* if it is well-formed and for each of its subterms of the form $t \uparrow s$ or $t \star s$ it is true that $s$ is of the form $c$ or $c^{-1}$, where $c$ is a pure, ground term;
- *$C$-exponent-ground* if it is exponent-ground and has exponents only from the predefined finite set $C$;
- *exponent-reduced* if all its subterms in an exponent position (right argument of $\star$ or $\uparrow$) are reduced.

All these notions are also lifted to atoms, clauses and theories in the natural way.

We need to define a $C$-exponent ground theory $T_E^C$ that allows multiplication and exponentiation only with ground multipliers (exponents). Let $C$ be the set of pure, ground terms that may be used as multipliers or exponents. The theory contains the following rules:

1) $I(x), I(y) \to I(e(x,y))$;
2) $I(x), I(c) \to I(x \uparrow c)$ for each $c \in C$;
3) $I(x), I(c) \to I(x \uparrow c^{-1})$ for each $c \in C$;
4) $I(x), I(c) \to I(x \star c)$ for each $c \in C$;
5) $I(x), I(c) \to I(x \star c^{-1})$ for each $c \in C$.

Here "for each $c \in C$" implies that the number of intruder rules is linearly dependent on the size of the set $C$.

In these rules we no longer need the rule that allows the intruder to find inverses of the elements. We do not need the inverses of group elements if we are dealing only with products in exponents, and the rules 3 and 5 actually give the intruder the ability to find inverses of integers when performing multiplication (exponentiation).

Let $T$ be a $C$-exponent-ground theory that represents some protocol. We need to show that if a $C$-exponent-ground atom $a$ can be derived using the properties of bilinear pairings from a $C$-exponent-ground theory $T$ (denote it as $T \cup T_E \vdash_\sim a$), then there exists a $C$-exponent-ground derivation of $a$.

**Theorem 1.** *Let $C$ be a set of pure, ground terms. Let $T$ be a $C$-exponent-ground Horn theory and $a$ be a $C$-exponent-ground atom. If $T \cup T_E \vdash_\sim a$, then there exists a $C$-exponent-ground derivation for $T \cup T_E^C \vdash_\sim a$, where the substitutions applied to this derivation are also $C$-exponent-ground.*

In order to prove Theorem 1, we need to show how to transform arbitrary derivation *modulo* $\sim$ to a $C$-exponent-ground derivation. We start by defining a function $\delta_C$ that turns any term into a $C$-exponent-ground term. Let $C^{-1} = \{c^{-1} | c \in C\}$, and let $C^* = C \cup C^{-1}$. The function $\delta_C$ is defined inductively:

$$\begin{aligned}
\delta_C(x) &= x && \text{for a variable or name } x \\
\delta_C(t \uparrow s) &= \delta_C(t) \uparrow s && \text{if } s \in C^* \\
\delta_C(t \uparrow s) &= \delta_C(t) && \text{if } s \notin C^* \\
\delta_C(t^{-1}) &= \delta_C(t) \\
\delta_C(t \star s) &= \delta_C(t) \star s && \text{if } s \in C^* \\
\delta_C(t \star s) &= \delta_C(t) && \text{if } s \notin C^* \\
\delta_C(f(t_1, \ldots t_n)) &= f(\delta_C(t_1), \ldots \delta_C(t_n)) && \text{for } f \notin \{\uparrow, \star, ^{-1}\}
\end{aligned}$$

This function throws away all the non-ground multipliers (exponents) and gets rid of the $^{-1}$ operation. We will show now that applying this function to some derivation $T \cup T_E \vdash_\sim a$ returns a $C$-exponent-ground derivation $T \cup T_E^C \vdash_\sim a$.

**Lemma 2.** *For any set C of pure, ground terms and for every term t we have:*

*1) $\delta_C(t)$ is $C$-exponent-ground.*

*2)* $\delta_C(t) = t$ *iff* $t$ *is* $C$*-exponent-ground.*
*3)* $\delta_C(\delta_C(t)) = \delta_C(t)$.

This lemma summarizes the properties of the function $\delta_C$. The proof of each point is based on induction. We have to look through all the possible cases of application of $\delta_C$. See the full version of this paper [21] for the proof.

The lemma shows that the function $\delta_C$ indeed turns arbitrary terms into $C$-exponent-ground terms, and does not modify the terms that have already been $C$-exponent-ground.

We need to show that if any two terms have been equivalent before, they still remain equivalent after applying the function $\delta_C$ to them. We will consider only the terms that are already exponent-reduced. It is necessary to prove that $\delta_C$ preserves the equivalence on exponent-reduced terms.

**Lemma 3.** *For any set* $C$ *of pure, ground terms and for all exponent-reduced terms* $t$ *and* $s$*, if* $t \sim s$*, then* $\delta_C(t) \sim \delta_C(s)$.

The proof, based on the induction over the size of $t$ and similar to [9], is given in [21].

**Example 1.** Let $C = \{c_1, c_2\}$. Let $t = enc(x^{-1} \star c_1, x \uparrow y \uparrow b)$, where $x, y$ are variables and $enc$ is some function.

$$\begin{aligned}
\delta_C(t) &= \delta_C(enc(x^{-1} \star c_1, x \uparrow y \uparrow c_2)) \\
&= enc(\delta_C(x^{-1} \star c_1), \delta_C(x \uparrow y \uparrow c_2)) \\
&= enc(\delta_C(x^{-1}) \star c - 1, \delta_C(x \uparrow y \uparrow c_2)) \\
&= enc(x \star c_1, \delta_C(x \uparrow y \uparrow c_2)) \\
&= enc(x \star c_1, \delta_C(x \uparrow y) \uparrow c_2) \\
&= enc(x \star c_1, \delta_C(x) \uparrow c_2) \\
&= enc(x \star c_1, x \uparrow c_2)
\end{aligned}$$

The function $\delta_C$ has in fact turned $x^{-1}$ and $x \uparrow y$ to $x$ (made these subterms $C$-exponent ground). Here we get the same term $x$ from different exponent-reduced terms $x^{-1}$ and $x$. This example shows why Lemma 3 works only in one direction. □

When ProVerif analyzes the protocol, the variables in terms are being substituted. We need to show that the function $\delta_C$ does not affect the substitution, and there is no difference if we apply $\delta_C$ before or after the substitution.

**Lemma 4.** *Let* $C$ *be a set of pure, ground terms. Let* $t$ *be a* $C$*-exponent-ground term, and* $\theta$ *be a substitution. Then* $\delta_C(t\theta) = t\delta_C(\theta)$.

Here $t\theta$ means that the substitution $\theta$ is applied to the term $t$, and $\delta_C(\theta)$ denotes applying the function $\delta_C$ to the terms that are going to substitute the variables of $t$. This lemma, an analogue of [9], is also proved in [21].

**Example 2.** Let $C = \{c_1, c_2\}$. Let $t = e(x, y) \uparrow c_1$, where $x, y$ are variables. Let $\theta = \{u^{-1}/x, v \star w/y\}$ be a substitution. In this example, it is not important if $e$ denotes pairing or it is some other function, since $\delta_C$ regards pairing in the same way like any other functions.

$$\begin{aligned}
\delta_C(t\theta) &= \delta_C((e(x, y) \uparrow c_1)\theta) \\
&= \delta_C(e(u^{-1}, v \star w) \uparrow c_1) \\
&= \delta_C(e(u^{-1}, v \star w)) \uparrow c_1 \\
&= e(\delta_C(u^{-1}), \delta_C(v \star w)) \uparrow c_1 \\
&= e(u, \delta_C(v \star w)) \uparrow c_1 \\
&= e(u, v) \uparrow c_1
\end{aligned}$$

$$\begin{aligned}
t\delta_C(\theta) &= t\delta_C(u^{-1}/x, v \star w/y) \\
&= t(\delta_C(u^{-1})/x, \delta_C(v \star w)/y) \\
&= t(u/x, v/y) \\
&= e(x, y) \uparrow c_1(u/x, v/y) \\
&= e(u, v) \uparrow c_1
\end{aligned}$$

It means that it does not matter whether we apply $\delta_C$ after the derivation in the end or apply it to the initial facts and only then start the unification process. □

**Sketch of proof for Theorem 1** According to the previous lemmas, we have that:

- $\delta_C$ turns any terms to $C$-exponent ground terms, and therefore it can be used to transform a non-$C$-exponent ground derivation to a $C$-exponent ground derivation.
- $\delta_C$ preserves equivalence on exponent-reduced terms.
- It does not matter whether we apply $\delta_C$ to $C$-exponent ground terms (including the terms of the initial $C$-exponent-ground theory $T$) before or after the substitution.

For any derivation step that uses rules from the initial $C$-exponent-ground theory $T$, we just need to apply $\delta_C$ to the substitution in order to ensure that we get $C$-exponent ground terms. The rules that belong to the theory $T_E$ (the intruder rules) are more complicated, but based on the previous lemmas it can be shown that we can use the intruder rules from $T_E^C$ instead of the rules from $T_E$. Again, the proof is very similar to [9], and it is given in the appendix. The proof is based on induction and case distinction for different kinds of rules.

## VI. Encoding of Terms

In this section we present an encoding of terms that hides most of the algebraic properties of bilinear pairings. The encoding is similar to [9], but more detailed. The main idea is to encode the terms in such a way that equivalent terms would have the same syntactical representation.

Let $C = \{c_1, \ldots, c_m\}$ be the set of pure, ground terms used in the derivation according to the theory $T$ using the signature $\Sigma$. Define $\Sigma^{pair} = (\Sigma \setminus \{\uparrow, ^{-1}, \star\}) \cup \{0, \mathsf{s}, \mathsf{p}, exp, mult\}$ as the new signature.

The constant $0$ and the unary functions $\mathsf{s}$ and $\mathsf{p}$ are used for encoding integers, as in [9]. The integer $n$ will be encoded as $\mathsf{s}^n(0) = \mathsf{s}(\ldots \mathsf{s}(0) \ldots)$, and $-n$ as $\mathsf{p}^n(0)$. This encoding defines two metatheoretical conversion functions $i2t(n)$ (integer to term) and $t2i(t)$ (term to integer).

The functions $mult$ and $exp$ are of arity $m+1$, and are used to encode multiplication in $G_1$ and exponentiation in $G_T$. The encoding of $C$-exponent-ground terms will be done over this signature. We need to consider only $C$-exponent-ground terms in the derivations. A term of the form $s \uparrow c_1^{(n_1)} \uparrow \ldots \uparrow c_m^{(n_m)}$ will be encoded as $exp(s, i2t(n_1), \ldots, i2t(n_m))$ over $\Sigma^{pair}$. Similarly, a term of the form $s \star c_1^{(n_1)} \star \ldots \star c_m^{(n_m)}$ will be encoded as $mult(s, i2t(n_1), \ldots, i2t(n_m))$.

**Example 3.** Let $C = \{c_1, c_2, c_3\}$. Let $t = e(x, y) \uparrow c_1 \uparrow c_1 \uparrow c_3^{-1}$.
The term $t$ will be encoded as $exp(e(x, y), \mathsf{s}^2(0), 0, \mathsf{p}(0))$. □

There are two more metatheoretical functions that have been defined for increasing and decreasing integers: $incr(t) = i2t(t2i(t) + 1)$ and $decr(t) = i2t(t2i(t) - 1)$. Formally, they are defined:

- $incr(t) = t'$, if $t = \mathsf{p}(t')$, and $incr(t) = \mathsf{s}(t)$ otherwise;
- $decr(t) = t'$, if $t = \mathsf{s}(t')$, and $decr(t) = \mathsf{p}(t)$ otherwise.

For each $i \in \{1, \ldots, m\}$ we define the metatheoretical functions $incr_i^X$ and $decr_i^X$, where $X \in \{mult, exp\}$. Applying the function $incr_i^X$ to a term $t$ increases the power of the exponent (multiplier) $c_i$ in the term $t$ by 1, and $decr_i^X$ is the inverse of $incr_i^X$. Formally, these functions are defined as follows.

$$incr_i^X(X(t_0, \ldots, t_m)) = \begin{cases} t_0, \text{ if } t_i = \mathsf{p}(0) \text{ and } t_j = 0 \text{ for all } j \neq i \\ X(t_0, \ldots, t_{i-1}, incr(t_i), t_{i+1}, \ldots, t_m), \text{ otherwise} \end{cases}$$

$$decr_i^X(X(t_0, \ldots, t_m)) = \begin{cases} t_0, \text{ if } t_i = \mathsf{s}(0) \text{ and } t_j = 0 \text{ for all } j \neq i \\ X(t_0, \ldots, t_{i-1}, decr(t_i), t_{i+1}, \ldots, t_m), \text{ otherwise.} \end{cases}$$

If $t$ is not of the form $X(t_0, \ldots, t_m)$ then

$$incr_i^X(t) = incr_i^X(X(t, 0, \ldots, 0)) \text{ and}$$
$$decr_i^X(t) = decr_i^X(X(t, 0, \ldots, 0)) \ .$$

**Example 4.** Let $C = \{c_1, c_2, c_3\}$. Then $m = |C| = 3$. Let $x, y$ be variables.

- $incr_2^{exp}(exp(e(x, y), 0, \mathsf{p}(0), 0)) = exp(e(x, y), 0, 0, 0)$;
- $incr_1^{mult}(x) = mult(x, \mathsf{s}(0), 0, 0)$;
- $decr_1^{exp}(exp(y, \mathsf{s}(\mathsf{s}(0)), 0, 0)) = exp(y, \mathsf{s}(0), 0, 0)$. □

The transformation of a $C$-exponent-ground term $t$ to a term $\ulcorner t \urcorner$ over $\Sigma^{pair}$ is given below.

- $\ulcorner x \urcorner = x$ for a variable or name $x$;
- $\ulcorner f(t_1, \ldots, t_n) \urcorner = f(\ulcorner t_1 \urcorner, \ldots, \ulcorner t_n \urcorner)$ $(f \notin \{\uparrow, ^{-1}, \star, e\})$;
- $\ulcorner t \uparrow c_i \urcorner = incr_i^{exp}(\ulcorner t \urcorner)$;
- $\ulcorner t \uparrow c_i^{-1} \urcorner = decr_i^{exp}(\ulcorner t \urcorner)$;
- $\ulcorner t \star c_i \urcorner = incr_i^{mult}(\ulcorner t \urcorner)$;
- $\ulcorner t \star c_i^{-1} \urcorner = decr_i^{mult}(\ulcorner t \urcorner)$;
- $\ulcorner e(t_1 \star c_i, t_2) \urcorner = \ulcorner e(t_1, t_2) \uparrow c_i \urcorner$;
- $\ulcorner e(t_1, t_2 \star c_i) \urcorner = \ulcorner e(t_1, t_2) \uparrow c_i \urcorner$;
- $\ulcorner e(t_1, t_2) \urcorner = e(\ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner)$ (only if the two previous rules do not apply);
- $\ulcorner p(t) \urcorner = p(\ulcorner t \urcorner)$, for an atom $p(t)$.

We need to show that the function $\ulcorner \cdot \urcorner$ preserves equivalence on the encoded terms.

**Lemma 5.** *For $C$-exponent-ground terms $t$ and $s$, if $t \sim s$, then $\ulcorner t \urcorner \sim \ulcorner s \urcorner$.*

The proof of this lemma is similar to the proof of [9], and it can be found in [21]. The proof becomes more complex since there are additional definitions regarding the terms whose head symbol is pairing function $e$.

**Example 5.** Let $C = \{c_1\}$. Let $t = enc(e(g \star c_1, g \star c_1^{-1}), m)$ where $g$ and $m$ are some variables.

$$\begin{aligned} \ulcorner t \urcorner &= \ulcorner enc(e(g \star c_1, g \star c_1^{-1}), m) \urcorner \\ &= enc(\ulcorner e(g \star c_1, g \star c_1^{-1}) \urcorner, \ulcorner m \urcorner) \\ &= enc(\ulcorner e(g, g \star c_1^{-1}) \uparrow c_1 \urcorner, m) \\ &= enc(incr_1^{exp}(\ulcorner e(g, g \star c_1^{-1}) \urcorner), m) \\ &= enc(incr_1^{exp}(\ulcorner e(g, g) \uparrow c_1^{-1} \urcorner), m) \\ &= enc(decr_1^{exp}(incr_1^{exp}(\ulcorner e(g, g) \urcorner)), m) \\ &= enc(decr_1^{exp}(incr_1^{exp}(e(\ulcorner g \urcorner, \ulcorner g \urcorner))), m) \\ &= enc(decr_1^{exp}(incr_1^{exp}(e(g, g))), m) \\ &= enc(e(g, g), m) \end{aligned}$$ □

## VII. Derivation Rules for Encoded Terms

Given a theory $T$, we will now present the construction of a theory $T_C$, such that a derivation $\vdash_\sim$ according to theory $T \cup T_E$ is equivalent to an *almost* purely syntactic derivation $\vdash_c$ according to $T_C$. Formally, the derivation $\vdash_c$ is according to an equational theory that is generated by the single equation $e(x, y) = e(y, x)$. This theory is much simpler than $\sim$ and can be readily handled by ProVerif. The precise meaning of the equivalence of definitions is given by Theorem 7 below. Theory $T_C$ is generally similar to the one defined in [9], but contains significant new details for handling the algebraic properties of pairings. The clauses of the theory $T_C$ that do not depend on the clauses of $T$ are given in Fig. 1.

The rules (2)–(4) deal with integers: the intruder must be able to derive any integer term. The rules (5)–(8) enable the intruder to switch between $t$ and $exp(t, 0, \ldots, 0)$, between $t$ and $mult(t, 0, \ldots, 0)$.

If the intruder knows $c_i$, he is allowed to multiply (exponentiate) the term with $c_i^{(n)}$ for any integer $n$. This kind of reduction works better with ProVerif than just multiplying (exponentiating) a term with $c_i$ $n$ times. Given a term $exp(x, x_1, ..., x_m)$, the intruder can produce $exp(x, x_1, ..., \mathsf{s}(x_i), ..., x_m)$ by exponentiating with $c_i$. Hence he can non-deterministically change the $i$-th counter to any other integer. The rule (9) deals with exponentiation, and the rule (10) deals with multiplication.

At this point we start diverging from [9]. Namely, we must handle the addition of exponents. The term $e(x \star c^{(x_1)} \star \ldots \star c^{(x_m)}, y \star c^{(y_1)} \star \ldots \star c^{(y_m)})$ is equivalent to $e(x, y) \uparrow c^{(z_1)} \uparrow \ldots \uparrow c^{(z_m)}$, where for each $i$: $z_i = x_i + y_i$; these terms have the same encoding. We handle the addition by introducing

$$
\begin{aligned}
& & & I(0) && (2)\\
I(x) & \rightarrow & & I(\mathsf{s}(x)) && (3)\\
I(x) & \rightarrow & & I(\mathsf{p}(x)) && (4)\\
I(x) & \rightarrow & & I(exp(x,0,\ldots,0)) && (5)\\
I(exp(x,0,\ldots,0)) & \rightarrow & & I(x) && (6)\\
I(x) & \rightarrow & & I(mult(x,0,\ldots,0)) && (7)\\
I(mult(x,0,\ldots,0)) & \rightarrow & & I(x) && (8)\\
I(c_i), I(y), I(exp(x_0,x_1,\ldots,x_m)) & \rightarrow & & I(exp(x_0,\ldots,x_{i-1},y,x_{i+1},\ldots,x_m)) && (9)\\
I(c_i), I(y), I(mult(x_0,x_1,\ldots,x_m)) & \rightarrow & & I(mult(x_0,\ldots,x_{i-1},y,x_{i+1},\ldots,x_m)) && (10)
\end{aligned}
$$

$$
\begin{aligned}
& & & INCR(x, incr(x)) \text{ for } x \in \{0, \mathsf{s}(y), \mathsf{p}(y)\} && (11)\\
& & & DECR(x, decr(x)) \text{ for } x \in \{0, \mathsf{s}(y), \mathsf{p}(y)\} && (12)\\
& & & A(x,0,x) && (13)\\
A(x,y,z), INCR(z,w) & \rightarrow & & A(x,\mathsf{s}(y),w) && (14)\\
A(x,y,z), DECR(z,w) & \rightarrow & & A(x,\mathsf{p}(y),w) && (15)
\end{aligned}
$$

$$
A(x_1,y_1,z_1),\ldots,A(x_m,y_m,z_m), I(mult(x,x_1,\ldots,x_m)), I(mult(y,y_1,\ldots,y_m)) \rightarrow I(exp(e(x,y),z_1,\ldots,z_m)) \quad (16)
$$

$$
\begin{aligned}
& E(t, c_i, incr_i^{exp}(t)) && \text{for each } c_i \in C \text{ and } t \in \{x, exp(x_0,\ldots,x_m), exp(x_0,\ldots,\mathsf{p}(x_i),\ldots,x_m), \ulcorner x \uparrow c_i^{-1} \urcorner\} && (17)\\
& E(t, c_i, decr_i^{exp}(t)) && \text{for each } c_i \in C^{-1} \text{ and } t \in \{x, exp(x_0,\ldots,x_m), exp(x_0,\ldots,\mathsf{s}(x_i),\ldots,x_m), \ulcorner x \uparrow c_i \urcorner\} && (18)\\
& M(t, c_i, incr_i^{mult}(t)) && \text{for each } c_i \in C \text{ and } t \in \{x, mult(x_0,\ldots,x_m), mult(x_0,\ldots,\mathsf{p}(x_i),\ldots,x_m), \ulcorner x \star c_i^{-1} \urcorner\} && (19)\\
& M(t, c_i, decr_i^{mult}(t)) && \text{for each } c_i \in C^{-1} \text{ and } t \in \{x, mult(x_0,\ldots,x_m), mult(x_0,\ldots,\mathsf{s}(x_i),\ldots,x_m), \ulcorner x \star c_i \urcorner\} && (20)
\end{aligned}
$$

$$
\begin{aligned}
& & & P(x,y,e(x,y)) && (21)\\
& & & P(mult(x,x_1,\ldots,x_m), y, exp(e(x,y),x_1,\ldots,x_m)) && (22)\\
& & & P(x, mult(y,y_1,\ldots,y_m), exp(e(x,y),y_1,\ldots,y_m)) && (23)\\
A(x_1,y_1,z_1),\ldots,A(x_m,y_m,z_m) & \rightarrow & & P(mult(x,x_1,\ldots,x_m), mult(y,y_1,\ldots,y_m), exp(e(x,y),z_1,\ldots,z_m)) && (24)\\
A(x_1,y_1,0),\ldots,A(x_m,y_m,0) & \rightarrow & & P(mult(x,x_1,\ldots,x_m), mult(y,y_1,\ldots,y_m), e(x,y)) && (25)
\end{aligned}
$$

Fig. 1. Generic clauses of the theory $T_C$

a predicate $A$. Metatheoretically, $A(x,y,z)$ is true iff $z = i2t(t2i(x) + t2i(y))$.

The predicate $A$ cannot be defined through case enumeration. We define it recursively, using auxiliary predicates $INCR$ and $DECR$. These definitions are expressed by the rules (11) — (15).

With help of the predicate $A$ we can describe the intruder's ability to compute the pairing of two terms by introducing the rule (16).

Similarly to [9] we define predicates $E$, $M$, and $P$ that will express exponentiation, multiplication and pairing for $C$-exponent-ground terms. Metatheoretically,

- $E(x,y,z)$ is true iff $x \uparrow y \sim z$.
- $M(x,y,z)$ is true iff $x \star y \sim z$.
- $P(x,y,z)$ is true iff $e(x,y) \sim z$.

The main purpose of these predicates is to bring terms to normal form, so that two terms are equivalent *modulo* $T_C$ iff they are syntactically equivalent *modulo* $e(x,y) = e(y,x)$. This allows ProVerif to unify equivalent terms without using the other equations of $T_C$. In the original protocol (the theory $T$), all the terms of the form $x \uparrow y$, $x \star y$, and $e(x,y)$ will be replaced with the corresponding terms $z$ that are equivalent according to the definitions of the predicates $E$, $M$, and $P$.

Predicates $E$ and $M$ are simple to express in theory $T_C$. The predicate $E$ has already been defined in [9], and $M$ is defined analogously. The rules (17) and (18) deal with exponentiation, and the rules (19) and (20) deal with multiplication.

The rules for bilinear mappings are a little bit longer, because we have to add multipliers for each $c_i \in C$ separately. This is the main reason why derivation time with ProVerif grows so rapidly with increasing the set $C$. The rules (21)—(23) refer to the simpler part of the definition of $P$, where at most one of the paired terms has $mult$ as its head operation. We do not need to use addition of exponents in these rules.

If we define in the same way the case where the head operation of both arguments of the pairing function is $mult$, we would get an infinite number of clauses. We need to describe this case in another way, using the auxiliary predicate

$A$ that we have already defined above. We do it by introducing the rule (24).

This rule alone is still not enough for the full description of the predicate $P$. It may give us answers like $exp(e(x,y),0,\ldots,0)$. Metatheoretically, the terms $exp(e(x,y),0,\ldots,0)$ and $e(x,y)$ denote the same quantity, but $exp(e(x,y),0,\ldots,0) \neq e(x,y)$. It is not a problem for the intruder rules since he has rules for transforming $exp(e(x,y),0,\ldots,0)$ to $e(x,y)$, but some congruences within the initial protocol $T$ would be lost. For example, there would be no syntactical equivalence between $e(x,y) \uparrow c_1^{-1} \uparrow c_1 = e(x,y)$ and $e(x \star c_1, y \star c_1^{-1}) = exp(e(x,y),0,\ldots,0)$. Therefore, we need to define a separate rule for this case — the rule (25).

Finally, we describe how the rules of the theory $T$ are encoded as rules in theory $T_C$. This encoding is again similar to [9]. ProVerif is able to handle the encoded rules without difficulty.

Any clause $r_1,\ldots,r_n \to r_0$ from a theory $T$ is encoded by substituting all non-ground, non-standard subterms with their $C$-exponent-ground encodings. Similarly to [9], the encoded clause is

$$\ulcorner\theta(r_1)\urcorner,\ldots,\ulcorner\theta(r_n)\urcorner,\mathcal{C} \to \ulcorner\theta(r_0)\urcorner, \tag{26}$$

where $\theta$ is the substitution from non-ground non-standard subterms to new variables, and $\mathcal{C}$ is a set of clauses establishing that these variables equal the subterms they've replaced. Formally, let $\mathcal{R}$ be the set of all subterms of $r_0,\ldots,r_n$ of the form $s \star c$, $s \uparrow c$ or $e(s_1,s_2)$, where $c \in C^*$ and $s$, $s_1$ or $s_2$ is non-ground. For each term $t \in \mathcal{R}$ let $x_t$ be a new variable. The substitution $\theta$ works in a top-down manner:

$$\begin{aligned}
\theta(u) &= u && \text{for name or variable } u\\
\theta(t) &= x_t && \text{if } t \in \mathcal{R}\\
\theta(f(t_1,\ldots,t_n)) &= f(\theta(t_1),\ldots,\theta(t_n)) && \text{otherwise}\\
\theta(p(t_1,\ldots,t_n)) &= p(\theta(t_1),\ldots,\theta(t_n)) && \text{for predicate } p.
\end{aligned}$$

The set of clauses $\mathcal{C}$ is defined as follows:

$$\begin{aligned}
\mathcal{C} \;=\; & \{M(\ulcorner\theta(s)\urcorner, c, x_{s\star c}) \mid s \star c \in \mathcal{R}\}\\
\cup\; & \{E(\ulcorner\theta(s)\urcorner, c, x_{s\uparrow c}) \mid s \uparrow c \in \mathcal{R}\}\\
\cup\; & \{P(\ulcorner\theta(s_1)\urcorner, \ulcorner\theta(s_2)\urcorner, x_{e(s_1,s_2)}) \mid e(s_1,s_2) \in \mathcal{R}\}\ .
\end{aligned}$$

We see that the substitution $\theta$ and encoding $\ulcorner\cdot\urcorner$ are applied also in the arguments of predicates $M$, $E$, $P$. In this manner more complex expressions involving the operations $\star, \uparrow, e$ can be encoded.

The following lemma states that if an instance of normalization predicates $E, P, M$ is defined correctly in $T$ (its metatheoretical meaning holds), then it can be derived in $T_C$.

**Lemma 6.** *Let $t$ and $s$ be $C$-exponent-ground terms, $c \in C \cup C^{-1}$. Then $E(\ulcorner t\urcorner, c, \ulcorner t \uparrow c\urcorner)$, $M(\ulcorner t\urcorner, c, \ulcorner t \star c\urcorner)$, $P(\ulcorner t\urcorner, \ulcorner s\urcorner, \ulcorner e(t,s)\urcorner)$ can be derived from the theory $T_C$.*

*Proof:*

We need this lemma in order to show that bringing the terms to normal form can indeed be performed by the rules of theory $T_C$. It shows that it can be done only for some particular uses of $E$, $M$, and $P$. For example, $E(t,s,s\star s)$ is not an instance of a fact of $T_C$. The further lemmas will show that we actually do not need it to hold for all possible cases. There are more cases that should be looked through compared to the analogous lemma in [9].

The proof is based on the definition of the encoding function $\ulcorner\cdot\urcorner$. In the rules, we may substitute the variables with any terms. Let $c \in C$, and let $t, s$ be $C$-exponent-ground terms.

- $E(\ulcorner t\urcorner, c, \ulcorner t \uparrow c\urcorner) = E(\ulcorner t\urcorner, c_i, incr_i^{exp}(\ulcorner t\urcorner)))$. This is an instance of the rule (17) for some $i$ where $c = c_i$.
- $E(\ulcorner t\urcorner, c^{-1}, \ulcorner t \uparrow c^{-1}\urcorner) = E(\ulcorner t\urcorner, c_i, decr_i^{exp}(\ulcorner t\urcorner)))$. This is an instance of (18) for some $i$ where $c = c_i$.
- $M(\ulcorner t\urcorner, c, \ulcorner t \star c\urcorner) = M(\ulcorner t\urcorner, c_i, incr_i^{mult}(\ulcorner t\urcorner)))$. This is an instance of (19) for some $i$ where $c = c_i$.
- $M(\ulcorner t\urcorner, c^{-1}, \ulcorner t \star c^{-1}\urcorner) = M(\ulcorner t\urcorner, c_i, decr_i^{mult}(\ulcorner t\urcorner)))$. This is an instance of (20) for some $i$ where $c = c_i$.
- If the head symbols of both $s$ and $t$ are not $\star$, then $P(\ulcorner t\urcorner, \ulcorner s\urcorner, \ulcorner e(t,s)\urcorner) = P(\ulcorner t\urcorner, \ulcorner s\urcorner, e(\ulcorner s\urcorner, \ulcorner t\urcorner))$, which is an instance of (21).

Let $\mathbf{C}$ be a multiset $\{\!|c_{i1},\ldots,c_{ik}|\!\}$ where each $c_{ij}$ is an element of $C \cup C^{-1}$. If $t$ is a term then let $t \star \mathbf{C}$ denote the term $t \star c_{i1} \star \cdots \star c_{ik}$ (it is well-defined up to $\sim$) and $t \uparrow \mathbf{C}$ denote the term $t \uparrow c_{i1} \uparrow \cdots \uparrow c_{ik}$. We also define the metatheoretical functions $incr^X_{\mathbf{C}}$, where $X \in \{mult, exp\}$, as follows:

$$\begin{aligned}
incr^X_{\emptyset}(t) &= t\\
incr^X_{\mathbf{C}\cup\{\!|c_i|\!\}}(t) &= incr^X_i(incr^X_{\mathbf{C}}(t)) && c_i \in C\\
incr^X_{\mathbf{C}\cup\{\!|c_i^{-1}|\!\}}(t) &= decr^X_i(incr^X_{\mathbf{C}}(t)) && c_i \in C\ .
\end{aligned}$$

Again, the functions $incr^{mult}_{\mathbf{C}}$ and $incr^{exp}_{\mathbf{C}}$ are well-defined due to the commutation properties of the functions $incr^X_i$ and $decr^X_i$. We use the defined notions to treat more complex cases of applying the predicate $P$.

- If the head symbol of $s$ is not $\star$, then $P(\ulcorner t \star \mathbf{C}\urcorner, \ulcorner s\urcorner, \ulcorner e(t,s) \uparrow \mathbf{C}\urcorner) = P(incr^{mult}_{\mathbf{C}}(\ulcorner t\urcorner), \ulcorner s\urcorner, incr^{exp}_{\mathbf{C}}(e(\ulcorner t\urcorner, \ulcorner s\urcorner)))$, which is an instance of (22), according to the definition of $incr^{exp}$ and $incr^{mult}$.
- If the head symbol of $t$ is not $\star$, then $P(\ulcorner t\urcorner, \ulcorner s \star \mathbf{C}\urcorner, \ulcorner e(t,s) \uparrow \mathbf{C}\urcorner) = P(\ulcorner t\urcorner, incr^{mult}_{\mathbf{C}}(\ulcorner s\urcorner), incr^{exp}_{\mathbf{C}}(e(\ulcorner t\urcorner, \ulcorner s\urcorner)))$, which is an instance of (23).
- $P(\ulcorner t \star \mathbf{C}_1\urcorner, \ulcorner s \star \mathbf{C}_2\urcorner, \ulcorner e(t,s) \uparrow \mathbf{C}_1 \uparrow \mathbf{C}_2\urcorner) = P(incr^{mult}_{\mathbf{C}_1}(\ulcorner t\urcorner), incr^{mult}_{\mathbf{C}_2}(\ulcorner s\urcorner), incr^{exp}_{\mathbf{C}_1\cup\mathbf{C}_2}(e(\ulcorner t\urcorner, \ulcorner s\urcorner)))$. Therefore it can be derived from (24).
- $P(\ulcorner t \star \mathbf{C}_1\urcorner, \ulcorner s \star \mathbf{C}_2\urcorner, \ulcorner e(t,s)\urcorner) = P(incr^{mult}_{\mathbf{C}_1}(\ulcorner t\urcorner), incr^{mult}_{\mathbf{C}_2}(\ulcorner s\urcorner), e(\ulcorner t\urcorner, \ulcorner s\urcorner))$. Hence it can be derived from the rule (25).

■

Let us consider the clauses given by the translation (26) for some clause $A = (r_1,\ldots,r_n \to r_0)$. Denote the clause in $T_C$ resulting from $A$ by $A^*$. First, if $A$ does contain neither multiplication, exponentiation nor the bilinear pairing

operations, then $A^* = A$. If $A$ contains some term $t \uparrow d$ with a non-ground term $t$, it is replaced by a fresh variable $y$, and the relation between $t$, $d$, and $y$ is captured by adding $E(t, d, y)$ to the clause. Similarly, a term $t \star d$ adds a new clause $M(t, d, y)$, and $e(s, t)$ adds a new clause $P(s, t, y)$. All terms are encoded using $\ulcorner . \urcorner$ to obtain terms over $\Sigma^{pair}$.

**Example 6.** Let $C = \{c_1, c_2\}$. Let *secret* be a constant of $T$. Suppose that $T$ contains the fact

$$R := I(e(x, y \star a)) \rightarrow I(secret).$$

For this clause, we get the rule

$$\begin{aligned} R' := \quad & M(\ulcorner\theta(y)\urcorner, c_1, z), P(\ulcorner\theta(x)\urcorner, \ulcorner\theta(y \star c_1)\urcorner, v), \\ & \ulcorner\theta(I(e(x, y \star c_1)))\urcorner \rightarrow \ulcorner\theta(I(secret))\urcorner. \end{aligned}$$

Here $z, v$ are newly introduced variables. They define the substitution $\theta = \{z/y \star c_1, v/e(x, y \star c_1)\}$. We get

$$R' = M(\ulcorner y\urcorner, c_1, z), P(\ulcorner x\urcorner, \ulcorner z\urcorner, v), \ulcorner I(v)\urcorner \rightarrow \ulcorner I(secret)\urcorner.$$

Since $x, y$ are variables, the function $\ulcorner . \urcorner$ does not do anything interesting, and we get

$$R' = M(y, c_1, z), P(x, z, v), I(v) \rightarrow I(secret).$$

Note that there are no instances of predicate $E$ since exponentiation symbol $\uparrow$ is not used in the initial clause.

Consider now the application of $R$ / $R'$ during the derivation. Let $\sigma = \{g \star c_1^{-1}/x, g \star c_2/y\}$ be the substitution that gives the values of $x$ and $y$ for which we want to apply the rule $R'$. If the derivation is being done according to the theory $T_C$, we have the substitution $\sigma' = \{mult(g, \mathsf{p}(0), 0)/x, mult(g, 0, \mathsf{s}(0))/y\}$ instead. Applying $\sigma'$ to $R'$ gives the instantiation

$$\begin{aligned} \sigma'(R') \equiv \quad & M(mult(g, 0, \mathsf{s}(0)), a, z), \\ & P(mult(g, \mathsf{p}(0), 0), z, v), \\ & I(v) \rightarrow \quad I(secret). \end{aligned}$$

The variables $z, v$ are auxiliary, and they remain free in $\sigma'(R)$. In order to establish the truth of the right hand side of this rule, the derivation engine (ProVerif) needs to find appropriate values for $z, v$. According to the definition of $M$ and $P$, there is only one way to evaluate them to true — $I(secret)$ will be established if $z = mult(g, \mathsf{s}(0), \mathsf{s}(0)))$ and $v = exp(e(g, g), 0, \mathsf{s}(0))$. In fact, we get the clause $I(exp(e(g, g), 0, \mathsf{s}(0))) \rightarrow I(secret)$ assuming that $M(\ldots)$ and $P(\ldots)$ are true. This rule belongs to $T_C$. □

**Theorem 7.** *Let $T$ be a non-trivial, $C$-exponent-ground theory over $\Sigma$ and $b = p(t)$ be a $C$-exponent-ground atom over $\Sigma$, with $p$ being a predicate occurring in $T$. Then, $T \cup T_E \vdash_\sim b$ iff $T_C \vdash_c \ulcorner b\urcorner$.*

If we prove this theorem, it means that any derivation *modulo* $\sim$ (using the properties of bilinear mappings) can be reduced to an almost purely syntactical derivation and can be analyzed by ProVerif. First, we need to prove several lemmas.

**Lemma 8.** *If there exists a $C$-exponent-ground derivation for $T \cup T_E^C \vdash_\sim b$ obtained using $C$-exponent-ground substitutions, then $T_C \vdash \ulcorner b\urcorner$.*

*Proof:* The proof of the lemma contains a larger number of different cases than the similar proof in [9]. It shows that using facts and rules encoded by $\ulcorner\cdot\urcorner$ one can derive all the $C$-exponent-ground atoms that can be derived by $T \cup T_E^C$, and the only difference is that the derived term will be encoded.

Let $\pi = b_1, \ldots, b_l$ be a $C$-exponent-ground derivation for $T \cup T_E^C \vdash_\sim b$ obtained using $C$-exponent-ground substitutions. The lemma can be proved by induction over the length of $\pi$:

- Base: If $l = 0$, there is no derivation
- Step: Let $\pi_{<l} = b_1, \ldots, b_{l-1}$. We know that $b \sim b_l$ can be derived from $\pi_{<l}$ by applying a clause from $T \cup T_E^C$ using a $C$-exponent-ground substitution $\sigma$. It is enough to show that $\ulcorner b\urcorner$ can be syntactically derived from $\ulcorner \pi_{<l}\urcorner$ using $T_C$. There are two cases to consider:

1) If $b$ is obtained using a clause of $T_E^C$, then $b = I(t)$ for some $C$-exponent-ground term $t$. There are three subcases:
   a) The set $\pi_{<l}$ contains atoms $I(r)$ for a $C$-exponent-ground $r$ and $I(c_i)$ for $c_i \in C$, such that $t \sim r \uparrow c_i$ or $t \sim r \uparrow c_i^{-1}$. The atom $I(\ulcorner t\urcorner)$ can be obtained from $I(\ulcorner r\urcorner)$ and $I(\ulcorner c_i\urcorner)$ using the following clauses:
      i) (5) if the reduced form of $r$ is standard.
      ii) (9) is used with an appropriate integer term derived by integer-derivation clauses (2)—(4).
      iii) If the reduced form of $t$ is standard, then (6) is applied.
   b) The set $\pi_{<l}$ contains atoms $I(r)$ for a $C$-exponent-ground $r$ and $I(c_i)$ for $c_i \in C$, such that $t \sim r \star c_i$ or $t \sim r \star c_i^{-1}$. The atom $I(\ulcorner t\urcorner)$ can be obtained from $I(\ulcorner r\urcorner)$ and $I(\ulcorner c_i\urcorner)$ using analogical clauses that are defined for multiplication in $T_C$.
   c) The set $\pi_{<l}$ contains atoms $I(r)$ and $I(s)$ for $C$-exponent-ground $r$ and $s$, such that $t \sim e(r, s)$. The atom $I(\ulcorner t\urcorner)$ can be obtained from $I(\ulcorner r\urcorner)$ and $I(\ulcorner s\urcorner)$ using the following clauses:
      i) If $r$(or $s$) is not of the form $mult(x_0, x_1, \ldots, x_m)$, then (7) must be applied to $r$(or $s$) in order to get $I(mult(r_0, r_1, \ldots, r_m))$ and $I(mult(s_0, s_1, \ldots, s_m))$, where $r \sim mult(r_0, r_1, \ldots, r_m)$ and $s \sim mult(s_0, s_1, \ldots, s_m)$.
      ii) Apply the rule (16) to $mult(r_0, r_1, \ldots, r_m)$ and $mult(s_0, s_1, \ldots, s_m)$.
2) If $b$ is obtained by some $C$-exponent-ground clause $r_1, \ldots, r_n \rightarrow r_0$ of $T$, there exists a $C$-exponent-ground substitution $\sigma$ such that $b \sim \sigma(r_0)$ and all $\sigma(r_1), \ldots, \sigma(r_n)$ belong to $\pi_{<l}$ (everything *modulo* $\sim$). The $\ulcorner b\urcorner$ can be obtained by using the clause that uses predicates $E$,$M$, and $P$. Denote the clause $r_1, \ldots, r_n \rightarrow r_0$ as $R \rightarrow S$. We will write out the rule (26):
   $M(\ulcorner\theta(s_1')\urcorner, b_1, x_1), \ldots, M(\ulcorner\theta(s_j')\urcorner, b_j, x_j)$,

$E(\ulcorner\theta(t'_1)\urcorner, d_1, y_1), \ldots, E(\ulcorner\theta(t'_k)\urcorner, d_k, y_k)$,
$P(\ulcorner\theta(u'_1)\urcorner, \ulcorner\theta(v'_1)\urcorner, z_1), \ldots, P(\ulcorner\theta(u'_l)\urcorner, \ulcorner\theta(v'_1)\urcorner, z_l)$,
$\ulcorner\theta(r_1)\urcorner, \ldots, \ulcorner\theta(r_n)\urcorner \to \ulcorner\theta(r_0)\urcorner$.

Define a substitution $\sigma^*$, which will be applied to $R \to S$ to obtain $\ulcorner b \urcorner$ as follows:

- $\sigma^*(x) = \ulcorner\sigma(x)\urcorner$, for $x \in var(r_1, \ldots, r_n)$;
- $\sigma^*(x_i) = \ulcorner\sigma(s_i)\urcorner$;
- $\sigma^*(y_i) = \ulcorner\sigma(t_i)\urcorner$;
- $\sigma^*(z_i) = \ulcorner\sigma(w_i)\urcorner$.

It is easy to show by induction that, for each subterm $u$ of $r_0, \ldots, r_m$, which is not of the form $w^{-1}$, we have $\sigma^*(\ulcorner\theta(u)\urcorner) = \ulcorner\sigma(u)\urcorner$:

a) If $u$ is standard, the claim immediately follows by induction hypothesis.

b) if $u$ is a ground, non-standard subterm, then both $\sigma^*(\ulcorner\theta(u)\urcorner)$ and $\ulcorner\sigma(u)\urcorner$ are equal to $\ulcorner u \urcorner$.

c) If $u$ is non-ground and non-standard, then:

i) if $u \in \{s_1, \ldots, s_k\}$, then $\theta(u) = x_i$ for some $i$.

ii) if $u \in \{t_1, \ldots, t_k\}$, then $\theta(u) = y_i$ for some $i$.

iii) if $u \in \{w_1, \ldots, w_k\}$, then $\theta(u) = z_i$ for some $i$.

The claim follows from the definition of $\sigma^*$.

Now we have that $\sigma^*(\ulcorner\theta(r_i)\urcorner) = \ulcorner\sigma(r_i)\urcorner$ for each $i \in \{0, \ldots, n\}$; in particular, $\sigma^*(\ulcorner\theta(r_i)\urcorner) \in \ulcorner\pi_{<l}\urcorner$ for each $i \in \{1, \ldots, n\}$, and we obtain $\sigma^*(\ulcorner\theta(r_i)\urcorner) = \ulcorner\sigma(r_0)\urcorner = \ulcorner b \urcorner$ by applying $R \to S$ with substitution $\sigma^*$ (the equality follows from Lemma 5). It remains to prove that:

$$\begin{aligned} E^* &:= \sigma^*(E(\ulcorner\theta(t'_i)\urcorner, d_i, y_i)), \\ M^* &:= \sigma^*(M(\ulcorner\theta(s'_i)\urcorner, b_i, x_i)), \\ P^* &:= \sigma^*(P(\ulcorner\theta(u'_i)\urcorner, \ulcorner\theta(v'_i)\urcorner, z_i)) \end{aligned}$$

can all be derived from $T_C$.

- For exponentiation, we have: $E^* = \sigma^*(E(\ulcorner\theta(t'_i)\urcorner, d_i, y_i)) = E(\sigma^*(\ulcorner\theta(t'_i)\urcorner), d_i, \sigma^*(y_i)) = E(\ulcorner\sigma(t'_i)\urcorner, d_i, \ulcorner\sigma(t_i)\urcorner) = E(\ulcorner\sigma(t'_i)\urcorner, d_i, \ulcorner\sigma(t'_i) \uparrow d_i \urcorner)$. By Lemma 6, the last fact is an instance of (17) or (18), depending on whether $d_i$ belongs to $C$ or $C^{-1}$.
- Analogically, for multiplication we have: $M^* = M(\ulcorner\sigma(s'_i)\urcorner, b_i, \ulcorner\sigma(s'_i) \star b_i \urcorner)$. By Lemma 6, this fact is an instance of (19) or (20), depending on whether $b_i$ belongs to $C$ or $C^{-1}$.
- For pairing, we have $P^* = \sigma^*(P(\ulcorner\theta(u'_i)\urcorner, \ulcorner\theta(v'_i)\urcorner, z_i)) = P(\sigma^*(\ulcorner\theta(u'_i)\urcorner), \sigma^*(\ulcorner\theta(v'_i)\urcorner), \sigma^*(z_i)) = P(\ulcorner\sigma(u'_i)\urcorner, \ulcorner\sigma(v'_i)\urcorner, \ulcorner\sigma(w_i)\urcorner) = P(\ulcorner\sigma(u'_i)\urcorner, \ulcorner\sigma(v'_i)\urcorner, \ulcorner e(\sigma(u'_i), \sigma(v'_i))\urcorner)$. By Lemma 6, this fact is an instance of:
  - (21) if the head symbol of both $u'_i$ and $v'_i$ is not $\star$.
  - (22) if the head symbol of $u'_i$ is $\star$, and the head symbol of $v'_i$ is not $\star$.
  - (23) if the head symbol of $v'_i$ is not $\star$, and the head symbol of $u'_i$ is $\star$.
  - (24) if the head symbols of both $u'_i$ and $v'_i$ are $\star$.
  - (25) if the head symbols of both $u'_i$ and $v'_i$ are $\star$, and after the pairing all the multipliers are going to be cancelled out ($A(x_1, y_1, 0), \ldots, A(x_m, y_m, 0)$ are true).

■

Theorem 1 states that the existence of derivation $T \cup T_E \vdash_\sim b$ implies the existence of derivation $T \cup T_E^C \vdash_\sim b$. Together with Lemma 8, this proves one direction of Theorem 7.

We need to show that the other direction also holds, thus establishing the soundness of the reduction. The next section describes decoding of the terms.

## VIII. Decoding the Terms: Soundness of the Reduction

We have proved that $T \cup T_E \vdash_\sim b$ implies $T_C \vdash_{\mathrm{c}} \ulcorner b \urcorner$, but this is not sufficient. Now we are going to prove that $T_C \vdash_{\mathrm{c}} \ulcorner b \urcorner$ implies $T \cup T_E \vdash_\sim b$. It is very important since it proves the soundness of our reduction. In order to do that, we need to define a decoding function that would turn terms over $\Sigma^{pair}$ back into terms over $\Sigma$.

In the process of decoding, we will use the non-triviality of the protocol theory $T$: there exists some $u$ such that $T \cup T_E \vdash_\sim I(u)$. If $T$ was empty, the verification task would be trivial.

Previously, the function $t2i$ was defined only on integer terms. Now the domain of $t2i$ will be extended to all terms in the same way as it has been done in [9]. If the term is not an integer term, then $t2i$ turns it to 0.

- $t2i(0) = 0$;
- $t2i(\mathsf{s}(t)) = t2i(t) + 1$;
- $t2i(\mathsf{p}(t)) = t2i(t) - 1$;
- $t2i(t) = 0$, for any term $t \notin \{0, \mathsf{s}(t'), \mathsf{p}(t')\}$ for some term $t'$.

Now we can define the decoding function, a mapping $\llcorner\cdot\lrcorner$ from terms over $\Sigma^{pair}$ to terms over $\Sigma$.

- $\llcorner x \lrcorner = x$, for a variable $x$;
- $\llcorner 0 \lrcorner = u$;
- $\llcorner \mathsf{s}(t) \lrcorner = u$;
- $\llcorner \mathsf{p}(t) \lrcorner = u$;
- $\llcorner exp(t, s_1, \ldots, s_m) \lrcorner = \llcorner t \lrcorner \uparrow c_1^{t2i(s_1)} \uparrow \ldots \uparrow c_m^{t2i(s_m)}$;
- $\llcorner mult(t, s_1, \ldots, s_m) \lrcorner = \llcorner t \lrcorner \star c_1^{t2i(s_1)} \star \ldots \star c_m^{t2i(s_m)}$;
- $\llcorner f(t_1, \ldots, t_n) \lrcorner = f(\llcorner t_1 \lrcorner, \ldots, \llcorner t_n \lrcorner)$, where $f \notin \{0, \mathsf{s}, \mathsf{p}, exp, mult\}$;
- $\llcorner p(t) \lrcorner = p(\llcorner t \lrcorner)$, for an atom $p(t)$.

In this definition, we need $u$ only to express that if an integer term occurs somewhere outside of multiplication or exponentiation, then its decoding can also be derived from $T$. The decoding is meaningless for the actual protocol, and is only necessary for the formal statement of the further lemmas: since the intruder can derive any integer term in $T_C$, we need to show that he can derive the same term in its decoded form. This has been done in the same way in [9].

**Example 7.** Let $C = \{c_1, c_2\}$. Suppose that we are given a term $t := exp(f(\mathsf{p}(0), x), 0, \mathsf{s}(\mathsf{s}(0)))$ over $\Sigma^{pair}$, where $x$ is a

variable and $f$ is arbitrary functional symbol. We have:

$$\begin{aligned}\llcorner t \lrcorner &= \llcorner exp(f(\mathsf{p}(0), x), 0, \mathsf{s}(\mathsf{s}(0))) \lrcorner \\ &= \llcorner f(\mathsf{p}(0), x) \lrcorner \uparrow c_2^2 \\ &= f(\llcorner \mathsf{p}(0) \lrcorner, \llcorner x \lrcorner) \uparrow c_2^2 \\ &= f(u, x) \uparrow c_2^2 \end{aligned}$$ □

There must be a relationship between the functions $\ulcorner . \urcorner$ and $\llcorner . \lrcorner$. Everything that was encoded may be later decoded. These functions are not the inverse functions of each other because syntactically different, but congruent terms have the same encoding. But we do not need the syntactical equality of the terms $t$ and $\llcorner \ulcorner t \urcorner \lrcorner$, equivalence *modulo* $\sim$ is sufficient.

**Lemma 9.** *Let $t$ be a $C$-exponent-ground term over $\Sigma$. Then $\llcorner \ulcorner t \urcorner \lrcorner \sim t$.*

This lemma can be proved by structural induction over $t$. If $t$ is standard, the statement immediately follows by the induction hypothesis. If $t$ is not standard, let $t'$ be its reduced form. We have to look through three cases for a non-standard form: exponentiation, multiplication and pairing. The proof is similar to [9], but it includes additional cases for pairing. It can be found in [21].

The next lemma shows that if any instance of the predicates $E$,$M$, or $P$ can be derived from $T_C$, then this instance is defined correctly according to the the metatheoretical meaning of $E$,$M$,$P$ (it indeed represents exponentiation, multiplication, or pairing). There is an analogous lemma in [9] without the proof.

**Lemma 10.** *Let $t$,$d$, and $s$ be ground terms over $\Sigma^{pair}$. Let $C^* = C \cup C^{-1}$.*

- *If $E(t, d, s)$ can be derived from $T_C$, then $d \in C \cup C^{-1}$ and $\llcorner s \lrcorner \sim \llcorner t \lrcorner \uparrow d$.*
- *If $M(t, d, s)$ can be derived from $T_C$, then $d \in C \cup C^{-1}$ and $\llcorner s \lrcorner \sim \llcorner t \lrcorner \star d$.*
- *If $P(r, s, t)$ can be derived from $T_C$, then $\llcorner t \lrcorner \sim e(\llcorner r \lrcorner, \llcorner s \lrcorner)$.*

The proof of each point can be carried out by case distinction, and it can be found in [21].

**Example 8.** Let $C = \{c_1, c_2\}$. According to the definition of $T_C$, the fact

$$P(x, mult(y, y_1, y_2), exp(e(x, y), y_1, y_2)),$$

where $x, y, y_1, y_2$ are variables, is a fact of $T_C$.

Consider substitution

$$\sigma = \{g/x, g/y, \mathsf{s}(0)/y_1, g/y_2\}$$

for a constant $g$. Then,

$$P(g, mult(g, \mathsf{s}(0), g), exp(e(g, g), \mathsf{s}(0), g))$$

is an instance of the previous fact. We have that

$$\begin{aligned}\llcorner exp(e(g, g), \mathsf{s}(0), g) \lrcorner &= \llcorner e(g, g) \lrcorner \uparrow c_1^{t2i(\mathsf{s}(0))} \uparrow c_2^{t2i(g)} \\ &= e(\llcorner g \lrcorner, \llcorner g \lrcorner) \uparrow c_1^1 \uparrow c_2^0 \\ &= e(g, g) \uparrow c_1.\end{aligned}$$

$$\begin{aligned}e(\llcorner g \lrcorner, \llcorner mult(g, \mathsf{s}(0), g) \lrcorner) &= e(g, \llcorner g \lrcorner \star c_1^{t2i(\mathsf{s}(0))} \star c_2^{t2i(g)}) \\ &= e(g, g \star c_1^1 \star c_2^0) \\ &= e(g, g \star c_1) \\ &\sim e(g, g) \uparrow c_1.\end{aligned}$$ □

Now we need to show that each fact that can be derived in $T_C$, can also be derived in $T$ in its decoded form. We cannot say that the fact derived in $T$ will be exactly the same after decoding. As it was said before, there is no one-to-one correspondence between $\ulcorner . \urcorner$ and $\llcorner . \lrcorner$, but the facts derived from $T$ and $T_C$ will be equivalent *modulo* $\sim$.

**Lemma 11.** *Let $a = p(t)$ be an atom, such that $p$ occurs in $T$. Then, $T_C \vdash_c a$ implies $T \cup T_E \vdash_\sim \llcorner a \lrcorner$.*

*Proof:* We need to take the derivation of $a$ in $T_C$, remove all the atoms of the form $E(\ldots)$, $M(\ldots)$, and $P(\ldots)$, and replace all the remaining atoms $a_i$ by $\llcorner a_i \lrcorner$. The proof can be carried out by induction and case distinction, in the same way as it has been done in [9]. There are additional cases for pairing.

Let $\pi = a_1, \ldots, a_l$ be a derivation for $T_C \vdash_c a$. The proof proceeds by induction over the length of $\pi$. The induction base is $l = 0$, there is nothing to show. For the induction step, we need to show that $\llcorner a_l \lrcorner$ can be derived from $\llcorner \pi_{<l} \lrcorner$, where $\pi_{<l} = a_1, \ldots, a_{l-1}$, and $\llcorner \pi_{<l} \lrcorner$ is the sequence of atoms obtained from $\pi_{<l}$ by removing all atoms of the form $E(\ldots)$, $M(\ldots)$, and $P(\ldots)$, by replacing all the remaining atoms $a_i$ by $\llcorner a_i \lrcorner$.

By assumption, predicate symbols $E$, $M$, and $P$ do not occur in $T$. It suffices to consider different cases of how $a_l$ is obtained.

Here we list all the possible cases that we have to consider, including those that have been proved in [9].

1) If $a_l$ is obtained using the integer derivation rules (2)—(4), it must be of the form $I(0)$, $I(\mathsf{s}(t))$, or $I(\mathsf{p}(t))$. Therefore, $\llcorner a_l \lrcorner = u$, and we have $T \cup T_E \vdash_\sim I(u)$.
2) If $a_l$ is obtained using the rules (4)—(8), it is enough to note that $\llcorner t \lrcorner = \llcorner exp(t, 0, \ldots, 0) \lrcorner = \llcorner mult(t, 0, \ldots, 0) \lrcorner$.
3) If $a_l$ is obtained using the rule (9), the atom $a_l$ is of the form $I(exp(s_0, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_m))$, such that $I(exp(s_0, \ldots, s_m))$, $I(c_i)$, and $I(s_i')$ occur in $\pi_{<l}$. Set $b = I(exp(s_0, \ldots, s_m))$. Then, $\llcorner b \lrcorner = I(\llcorner exp(s_0, \ldots, s_m) \lrcorner)$ and $\llcorner I(c_i) \lrcorner = I(c_i)$ are elements of $\llcorner \pi_{<l} \lrcorner$.
   Now we need to derive $\llcorner a_l \lrcorner$ from $\llcorner b \lrcorner$ and $I(c_i)$.
   - If $t2i(s_i') > t2i(s_i)$, apply the clause $I(x), I(y) \to I(x \uparrow y)$ from $T_E$ $t2i(s_i') - t2i(s_i)$ times.
   - If $t2i(s_i') < t2i(s_i)$, apply the clause $I(x) \to I(x^{-1})$ to $I(c_i)$, then apply the rule $I(x), I(y) \to I(x \uparrow y)$ from $T_E$ $t2i(s_i) - t2i(s_i')$ times.
   - If $t2i(s_i') = t2i(s_i)$, then $\llcorner a_l \lrcorner$ is a repetition of $\llcorner b \lrcorner$.
4) If $a_l$ is obtained using the rule (10) the proof is analogical to exponentiation.
5) If $a_l$ is obtained using the rule: (16) then the atom $a_l$ is of the form $exp(e(u_0, v_0), s_1, \ldots, s_m)$,

such that $I(mult(u_0, u_1, \ldots, u_m))$ and $I(mult(v_0, v_1, \ldots, v_m))$ occur in $\pi_{<l}$, and the predicates $A(u_1, v_1, s_1), \ldots, A(u_m, v_m, s_m)$ are true. By definition of the predicate $A(\ldots)$, they are true iff $s_i = i2t(t2i(u_1) + t2i(v_i))$ for each $i$.

Now denote $b_1 = I(mult(u_0, u_1, \ldots, u_m))$,$b_2 = I(mult(v_0, v_1, \ldots, v_m))$. We get that $\llcorner b_1 \lrcorner = I(\llcorner mult(u_0, u_1, \ldots, u_m) \lrcorner)$ and $\llcorner b_2 \lrcorner = I(\llcorner mult(v_0, v_1, \ldots, v_m) \lrcorner)$ are elements of $\llcorner \pi_{<l} \lrcorner$. Then, $\llcorner a_l \lrcorner$ can be derived from $\llcorner b_1 \lrcorner$ and $\llcorner b_2 \lrcorner$ by applying the rule $I(x), I(y) \to I(e(x, y))$ from $T_E$.

We get a term of the form $e(\ldots, \ldots)$ without any exponents, but using equations $e(x, y \star z) = e(x, y) \uparrow z$ and $e(x, y) = e(y, x)$ we get that the result is actually equivalent to $\llcorner a_l \lrcorner$. Each exponent $c_i$ is taken out from $\llcorner b_1 \lrcorner$ $u_i$ times, and from $\llcorner b_2 \lrcorner$ $v_i$ times. As the result, we get $\llcorner exp(e(u_0, v_0), i2t(t2i(u_1)+t2i(v_1)), \ldots, i2t(t2i(u_m)+t2i(v_m))) \lrcorner = \llcorner exp(e(u_0, v_0), s_1, \ldots, s_m)) \lrcorner = \llcorner a_l \lrcorner$.

6) Suppose that $a_l$ is obtained using the rule (26). We write this rule out in more details:
$M(\ulcorner \theta(s'_1) \urcorner, b_1, x_1), \ldots, M(\ulcorner \theta(s'_j) \urcorner, b_j, x_j)$,
$E(\ulcorner \theta(t'_1) \urcorner, d_1, y_1), \ldots, E(\ulcorner \theta(t'_k) \urcorner, d_k, y_k)$,
$P(\ulcorner \theta(u'_1) \urcorner, \ulcorner \theta(v'_1) \urcorner, z_1), \ldots, P(\ulcorner \theta(u'_l) \urcorner, \ulcorner \theta(v'_1) \urcorner, z_l)$,
$\ulcorner \theta(r_1) \urcorner, \ldots, \ulcorner \theta(r_n) \urcorner \to \ulcorner \theta(r_0) \urcorner$.
Assume that this clause was instantiated with a substitution $\sigma$: it means that $a_l = \sigma(\ulcorner r_0 \urcorner)$. Furthermore, all the $\sigma(\ulcorner \theta(r_i) \urcorner)$ for all $i \in \{1, \ldots, n\}$, and all the $E(\sigma(\ulcorner \theta(t'_i) \urcorner), d_i, \sigma(y_i))$, $M(\sigma(\ulcorner \theta(s'_i) \urcorner), b_i, \sigma(x_i))$, and $P(\sigma(\ulcorner \theta(u'_i) \urcorner), \sigma(\ulcorner \theta(v'_i) \urcorner), \sigma(z_i))$ for all corresponding indices $i$, are in $\pi_{<l}$. Therefore, $\llcorner \sigma(\ulcorner \theta(r_i) \urcorner) \lrcorner$ for all $i \in \{1, \ldots, n\}$ are in $\llcorner \pi_{<l} \lrcorner$.
By Lemma 10, we have that $\llcorner \sigma(y_i) \lrcorner \sim \llcorner \sigma(\ulcorner \theta(t'_i) \urcorner) \lrcorner \uparrow d_i$, $\llcorner \sigma(x_i) \lrcorner \sim \llcorner \sigma(\ulcorner \theta(s'_i) \urcorner) \lrcorner \star b_i$, $\llcorner \sigma(z_i) \lrcorner \sim e(\llcorner \sigma(\ulcorner \theta(u'_i) \urcorner) \lrcorner, \llcorner \sigma(\ulcorner \theta(v'_i) \urcorner) \lrcorner)$.
Let $\sigma^*(x) = \llcorner \sigma(x) \lrcorner$. For each subterm $t$ of $r_0, \ldots, r_n$ such that $t$ is not of the form $w^{-1}$, we show by induction over the size of $t$, that $\sigma^*(t) \sim \llcorner \sigma(\ulcorner \theta(t) \urcorner) \lrcorner$:
   a) If $t = x$ is a variable: $\ulcorner \theta(x) \urcorner = x$, and thus $\sigma^*(x) = \llcorner \sigma(\ulcorner \theta(x) \urcorner) \lrcorner$, by definition of $\sigma^*$.
   b) If $t = f(t_1, \ldots, t_n)$ for $f \notin \{\uparrow, \star, e\}$: the claim easily follows by induction.
   c) If $t = t' \uparrow d$ and $t$ is ground: $\llcorner \sigma(\ulcorner \theta(t) \urcorner) \lrcorner = \llcorner \ulcorner t \urcorner \lrcorner$, and $\sigma^*(t) = t$. We know that $\llcorner \ulcorner t \urcorner \lrcorner \sim t$ by Lemma 9.
   d) If $t = t' \star d$ and $t$ is ground: similar to exponentiation.
   e) If $t = t_i = t'_i \uparrow d_i$:
   by the induction hypothesis, we have
   $\sigma^*(t_i) = \sigma^*(t'_i) \uparrow d_i \sim \llcorner \sigma(\ulcorner \theta(t'_i) \urcorner) \lrcorner \uparrow d_i$.
   We have $\llcorner \sigma(y_i) \lrcorner \sim \llcorner \sigma(\ulcorner \theta(t'_i) \urcorner) \lrcorner \uparrow d_i$. Therefore, $\sigma^*(t_i) \sim \llcorner \sigma(y_i) \lrcorner = \llcorner (\ulcorner \theta(t_i) \urcorner) \lrcorner$.
   f) If $t = s_i = s'_i \star b_i$: similar to exponentiation.
   g) If $t = w_i = e(u'_i, v'_i)$:
   by the induction hypothesis, we have
   $\sigma^*(w_i) = e(\sigma^*(u'_i), \sigma^*(v'_i))$
   $\sim e(\llcorner \sigma(\ulcorner \theta(u'_i) \urcorner) \lrcorner, \llcorner \sigma(\ulcorner \theta(v'_i) \urcorner) \lrcorner)$.
   We have $\llcorner \sigma(z_i) \lrcorner \sim e(\llcorner \sigma(\ulcorner \theta(u'_i) \urcorner) \lrcorner, \llcorner \sigma(\ulcorner \theta(v'_i) \urcorner) \lrcorner)$.
   Therefore, $\sigma^*(w_i) \sim \llcorner \sigma(z_i) \lrcorner = \llcorner (\ulcorner \theta(w_i) \urcorner) \lrcorner$.

By the above, we have that $\sigma^*(r_i) \sim \llcorner \sigma(\ulcorner \theta(r_i) \urcorner) \lrcorner$ ($r_i$ can not be of the form $w^{-1}$ since it is $C$-exponent-ground). We have that all the $\llcorner \sigma(\ulcorner \theta(r_i) \urcorner) \lrcorner$ for all $i \in \{1, \ldots, n\}$ are in $\llcorner \pi_{<l} \lrcorner$, which means that we can apply the clause $r_1, \ldots, r_n \to r_0$ with $\sigma^*$ to obtain $\sigma^*(r_0) \sim \llcorner \sigma(\ulcorner \theta(r_0) \urcorner) \lrcorner = \llcorner a_l \lrcorner$. ■

**Example 9.** Let $C = \{c_1, c_2\}$. Let *secret* be a constant of $T$. Let $f$ be an arbitrary functional symbol. Suppose that the atom $\widetilde{a} \equiv I(exp(e(g, g), 0, \mathsf{s}(0)))$ has been derived from $T_C$ using the rule

$$R' = I(f(x, y)), M(y, c_1, z), P(x, z, v) \to I(v).$$

Assume that $T_C$ contains a fact

$$I(f(mult(g, \mathsf{p}(0), 0), mult(g, 0, \mathsf{s}(0)))),$$

and the substitution obtained in the derivation process is

$$\begin{aligned} \sigma \;=\; \{&mult(g, \mathsf{p}(0), 0)/x, \\ &mult(g, 0, \mathsf{s}(0))/y, \\ &mult(g, \mathsf{s}(0), \mathsf{s}(0))/z, \\ &exp(e(g, g), 0, \mathsf{s}(0))/v\}. \end{aligned}$$

Note that after fixing $x$ or $y$, the value of $z$ is uniquely determined since otherwise the predicates $M(\ldots)$ and $P(\ldots)$ would be false, and the atom $\widetilde{a}$ could not have been obtained in this derivation.

Applying $\sigma$ and removing the instances of $M$ and $P$ from the derivation leaves the rules:

1) $I(f(mult(g, \mathsf{p}(0), 0), mult(g, 0, \mathsf{s}(0))))$;
2) $I(f(mult(g, \mathsf{p}(0), 0), mult(g, 0, \mathsf{s}(0)))) \to I(exp(e(g, g), 0, \mathsf{s}(0)))$.

After decoding the terms, we get:

1) $I(f(g \star c_1^{-1}, g \star c_2)) \to I(e(g, g) \uparrow c_2)$;
2) $I(f(g \star c_1^{-1}, g \star c_2))$.

These two rules belong to the theory $T$, and the clause $I(e(g, g) \uparrow c_2)$ can be obviously derived from $T$. On the other hand, $I(e(g, g) \uparrow c_2) \sim \llcorner \widetilde{a} \lrcorner$. □

**Proof of Theorem 7:** We have used Lemma 8 to prove that $T \cup T_E \vdash_\sim b$ implies $T_C \vdash_\mathrm{c} \ulcorner b \urcorner$. Now suppose that $T_C \vdash_\mathrm{c} \ulcorner b \urcorner$. By assumption, $b = p(t)$, where $p$ occurs in $T$. Lemma 11 implies that $T \cup T_E \vdash_\sim \llcorner \ulcorner b \urcorner \lrcorner$. By Lemma 9, $\llcorner \ulcorner b \urcorner \lrcorner \sim b$, and therefore $T \cup T_E \vdash_\sim b$. ■

As the result, we can say that instead of analyzing the $C$-exponent-ground protocol that uses bilinear pairings in $T \cup T_E$, it can be analyzed in $T \cup T_E^C$ without any loss of information.

## IX. Experiments

We have extended the Horn theory transformer by Küsters and Truderung [9] to also handle pairing operations. In their transformer, the protocol is written first as an ordinary Prolog program, and afterwards it is translated to a file that can be tested by ProVerif.

With the help of this extended transformer, we have used ProVerif to analyze several protocols employing bilinear pairings. All of them are key-agreement protocols. In our experiments, we have asked whether the attacker (an insider in the system) is capable of finding or determining the session key. Namely, at the end of the session, each party encrypts a secret value $sec$ with the key determined during the session, and releases it to the network. We query whether $I(sec)$ can be derived. The results of our tests are the following:

- **Joux's protocol**. This [5] is the original three-party Diffie-Hellman key exchange protocol without any authentication of messages. Our analysis finds it secure if the channels between parties are authenticated, and insecure otherwise.
- **A variation of Shim's protocol**. We consider the repaired version [22] of Shim's three-party certificate-based one-pass key-exchange protocol [23]. Our analysis finds it secure. There is an interesting detail in the model of this protocol. Namely, when a party receives the messages from other parties, it is supposed to verify whether two terms $t_1$ and $t_2$, both constructed by this party as $t_1 = e(t_{11}, t_{12})$ and $t_2 = e(t_{21}, t_{22})$, are equal. In symbolic model, obviously the equality *modulo* $\sim$, not the syntactic equality (supported natively by ProVerif) has to be used. We thus add the atoms $P(t_{11}, t_{12}, T)$ and $P(t_{21}, t_{22}, T)$ as premises to the clause corresponding to the protocol action that depends on the results of this comparison.
- **TAK 1**. This certificate-based three-party key-exchange protocol by Al-Riyami and Paterson [6] includes the public keys in the generated session key. It was found to be secure.
- **TAK 2**. This protocol by the same authors [6], although similar to the previous one, is insecure [24]. We were able to find the same attack using the theory transformer and ProVerif.
- **A Six Pass Pairing Based AKC Protocol**. This protocol is also proposed by Al-Riyami and Paterson [6]. It was found to be secure.

Our implementation is available from http://comserv.cs.ut.ee/forms/ati_report//index.php?year=2011.

*Efficiency*

We have tested the running time of the analysis on the listed protocols. Similarly to [9], our Horn theory transformer has negligible running time. Unfortunately, the situation is different for ProVerif's running time on transformed protocols. In our experiments, the running time of ProVerif seems to grow fast with the number of pairing operations in the protocols.

Our experiments were performed with ProVerif 1.84 on a 2.21 GHz AMD Athlon 64X2 Dual Core Processor 4400+ with 2GB of RAM. The running time for the simplest of the protocols — Joux's key exchange — was still a fraction of a second. In this protocol, each participant has to perform just a single pairing per session. TAK 1, six-pass AKC, and repaired Shim's protocol each required time between 6 and 23 minutes. TAK 2 protocol makes the most use of pairings (the agreed key in this protocol is the hash of the concatenation of results of three different pairing operations). The time to find the attack in this protocol amounted to several hours.

In our experiments, we did away with the commutation rule $e(x, y) \sim e(y, x)$. This means that ProVerif performed a purely syntactic derivation. This change was sound because in all protocols, only a single generator $p$ of the group $G_1$ was considered. This meant that $e(p, p)$ was the only considered generator for the group $G_T$. Our transformation $\ulcorner\cdot\urcorner$ thus brings all terms involving pairings to the form $exp(e(p, p), \ldots)$. Getting rid of the commutation rule improved the running time of ProVerif a couple of times, but the more complex protocols still required a long time to analyze.

## X. Addition in $G_1$ and multiplication in $G_T$

Our treatment allows neither the protocol participants nor the adversary to apply the full set of operations available in the groups $G_1$ and $G_T$ "in real life". Namely, similarly to previous approaches [9], we have very much concentrated on Diffie-Hellman-like protocols and exponentiation operations in groups. Addition in $G_1$ and multiplication in $G_T$ have not been a part of the signature, meaning that no-one could apply those. As these operations can interfere with existing operations in our signature (meaning: there are equations involving both of them and the exponentiation operations) we may ask whether an adversary capable of applying them could attack protocols that our analysis has found to be secure. This issue would not have arisen if there were no such interference [25].

We will now extend the signature $\Sigma$ with addition '$+$' in $G_1$ and multiplication '$\cdot$' in $G_T$, introducing the equations and intruder theory which make them behave as the corresponding operations in cyclic groups. If $\sim^\sharp$ is the new equivalence of terms, and $T_{E^\sharp}$ is the new intruder theory, then we show that for a protocol $P$ that does not contain operations of addition in $G_1$ and multiplication in $G_T$, and for an atom $a$ that similarly does not contain these operations, $T_P \cup T_{E^\sharp} \vdash_{\sim^\sharp} a$ implies $T_P \cup T_E \vdash_\sim a$. Hence these operations do not help the adversary.

The equations involving $+$ and $\cdot$ are given below. We additionally make use of two free constants 0 and 1, denoting the zero element in $G_1$ and the unit element in $G_T$.

$$\begin{array}{cc}
x + y \sim^\sharp y + x & x \cdot y \sim^\sharp y \cdot x \\
(x + y) + z \sim^\sharp x + (y + z) & (x \cdot y) \cdot z \sim^\sharp x \cdot (y \cdot z) \\
(x + y)^{-1} \sim^\sharp x^{-1} + y^{-1} & (x \cdot y)^{-1} \sim^\sharp x^{-1} \cdot y^{-1} \\
x + 0 \sim^\sharp x \quad x + x^{-1} \sim^\sharp 0 & x \cdot 1 \sim^\sharp x \quad x \cdot x^{-1} \sim^\sharp 1 \\
(x + y) \star z \sim^\sharp (x \star z) + (y \star z) & e(x + y, z) \sim^\sharp e(x, z) \cdot e(y, z) \\
(x \cdot y) \uparrow z \sim^\sharp (x \uparrow z) \cdot (y \uparrow z) &
\end{array} \tag{27}$$

The additions to the intruder theory due to these operations are straightforward. The theory $T_{E^\sharp}$ consists of the rules of the theory $T_E$, plus the following rules:

$$I(x), I(y) \to I(x + y) \qquad I(x), I(y) \to I(x \cdot y) \ .$$

As the protocol $P$ does not contain $+$ or $\cdot$, they can be introduced into a derivation only through adversarial rules.

We will show now that there is no reason for the adversary to introduce those operations — if the goal of the adversary is to establish an atom $a$ not containing $+$ or $\cdot$, then any derivation where $+$ and $\cdot$ are introduced can be repeated without their introduction. The necessary definitions for formally stating and establishing these results are given below, while their proofs are given in the appendix.

We will extend the definition of a *reduced* term to the terms that contain $+$ and $\cdot$ operations. We call a term over $\Sigma$ *reduced* if no equations in last four rows of (1) and last four rows of (27) can be applied to it from left to right *modulo* the equations in the first two rows of (1) and first two rows of (27). Hence in a reduced term, when several operations in and between $G_1$ and $G_T$ have to be performed, we first perform the pairings $e$, then the exponentiation-like operations $\star$ and $\uparrow$, then the inversions, and finally the multiplication-like operations $+$ and $\cdot$. Each term can be brought to a reduced form and the reduced form is determined uniquely *modulo* the associativity and/or commutativity of $\star$, $\uparrow$, $+$, $\cdot$ and $e$.

**Example 10.** Suppose that we are given a term

$$t := e((p \star a + p \star b) \star c, p \star a \star c) \uparrow a^{-1} \ .$$

We will bring the term $t$ to its reduced form:

$$\begin{aligned} t &= e((p \star a + p \star b) \star c, p \star a \star c) \uparrow a^{-1} \\ &\sim^\sharp e(p \star a \star c + p \star b \star c, p \star a \star c) \uparrow a^{-1} \\ &\sim^\sharp e(p \star a \star c + p \star b \star c, p) \uparrow a^{-1} \uparrow a \uparrow c \\ &\sim^\sharp e(p \star a \star c + p \star b \star c, p) \uparrow c \\ &\sim^\sharp (e(p \star a \star c, p) \cdot e(p \star b \star c, p)) \uparrow c \\ &\sim^\sharp e(p \star a \star c, p) \uparrow c \cdot e(p \star b \star c, p) \uparrow c \\ &\sim^\sharp e(p, p) \uparrow c \uparrow a \uparrow c \cdot e(p, p) \uparrow c \uparrow b \uparrow c \\ &\sim^\sharp e(p, p) \uparrow a \uparrow c^2 \cdot e(p, p) \uparrow b \uparrow c^2 \end{aligned}$$

In this example, the $+$ operation has disappeared from $t$ because it has been applied to terms under pairing. Note that the operation $+$ may still remain in the reduced form. The $+$ and $\cdot$ operations are also not restricted to head operations. They can be hidden inside the term, and they may remain there after the reduction. For example, $f(a+b) \uparrow c$ is a reduced term. □

According to this new definition of *reduced* form, the terms (and their subterms) that do not contain the operations $+$ and $\cdot$ are reduced in the same way as according to the previous definition (without $+$ and $\cdot$).

We define a function $DF$ ("decomposed form") that we will use to remove all the $+$ and $\cdot$ operations and replace them with single addends and factors. This can be done by defining special functions that will select particular elements from the sums and the products. Additionally, we need to ensure that applying these functions will keep equivalence *modulo* the equations of (27).

Let $\mathbf{Trm}^{\overline{+\cdot}}$ be the set of *reduced* terms whose head symbol is not $+$ or $\cdot$. Let $\mathcal{P}^{\neq\emptyset}_{\text{fin}}(X)$ denote the set of all the finite non-empty subsets of $X$. We define $\mathbf{Sel}^{\overline{+\cdot}}$ as the set of pairs of functions $(g^+, \dot{g})$ from $\mathcal{P}^{\neq\emptyset}_{\text{fin}}(\mathbf{Trm}^{\overline{+\cdot}})$ to $\mathbf{Trm}^{\overline{+\cdot}}$, satisfying the following conditions for all $X, Y \in \mathcal{P}^{\neq\emptyset}_{\text{fin}}(\mathbf{Trm}^{\overline{+\cdot}})$:

- $g^+(X) \in X$ and $\dot{g}(X) \in X$;
- $g^+(X \cup \{0\}) = g^+(X)$ and $\dot{g}(X \cup \{1\}) = \dot{g}(X)$;
- if $Y = \{x \star t | x \in X\}$, then $g^+(Y) = g^+(X) \star t$;
- if $Y = \{x \uparrow t | x \in X\}$, then $\dot{g}(Y) = \dot{g}(X) \uparrow t$;
- if $Y = \{e(x,t) | x \in X\}$, then $\dot{g}(Y) = e(g^+(X), t)$.

We use the functions $g^+$ and $\dot{g}$ to select addends from sums and factors from products. For a term $t$ in reduced form and a pair of functions $(g^+, \dot{g}) \in \mathbf{Sel}^{\overline{+\cdot}}$ we define a term $DF(t, g^+, \dot{g})$ as follows:

- $DF(u, g^+, \dot{g}) = u$
  (for a variable or a name $u$);
- $DF(f(t_1, \ldots, t_k), g^+, \dot{g}) =$
  $= f(DF(t_1, g^+, \dot{g}), \ldots, DF(t_k, g^+, \dot{g}))$
  (for $f \notin \{+, \cdot\}$);
- $DF(t_1 + \ldots + t_k, g^+, \dot{g}) = DF(g^+(\{t_1, \ldots, t_k\}), g^+, \dot{g})$
  (where the head symbol of $t_i$ is not $+$);
- $DF(t_1 \cdot \ldots \cdot t_k, g^+, \dot{g}) = DF(\dot{g}(\{t_1, \ldots, t_k\}), g^+, \dot{g})$
  (where the head symbol of $t_i$ is not $\cdot$).

We see that the term $DF(t, g^+, \dot{g})$ is a "simplified" form of $t$ in the sense that wherever in $t$ a sum or a product is contained, the term $DF(t, g^+, \cdot g)$ only contains a single addend or factor of this sum or product.
For an atom $a = P(t_1, \ldots, t_k)$ and functions $g^+, \dot{g}$ we define the new atom $DF(P(t_1, \ldots, t_k), g^+, \dot{g}) = P(DF(t_1, g^+, \dot{g}), \ldots, DF(t_k, g^+, \dot{g}))$.

**Example 11.** Suppose that we have a term

$$s := enc(p \star a + f(c \cdot d) \star b, msg) \ ,$$

the function $DF$ will decompose the reduced form of $s$ to addends:

$$\begin{aligned} s' &= DF(s, g^+, \dot{g}) \\ &= DF(enc(p \star a + f(c \cdot d) \star b, msg), g^+, \dot{g}) \\ &= enc(DF(p \star a + f(c \cdot d) \star b, g^+, \dot{g}), DF(msg, g^+, \dot{g})) \\ &= enc(DF(g^+(\{p \star a, f(c \cdot d) \star b\}), g^+, \dot{g}), msg) \end{aligned}$$

Depending on the definition of $g^+$, we select either the first or the second argument of addition.
If we select $p \star a$:

$$\begin{aligned} s' &= enc(DF(p \star a, g^+, \dot{g}), msg) \\ &= enc(DF(p, g^+, \dot{g}) \star a, msg) \\ &= enc(p \star a, msg) \end{aligned}$$

If we select $f(c \cdot d) \star b$:

$$\begin{aligned} s' &= enc(DF(f(c \cdot d) \star b, g^+, \dot{g}), msg) \\ &= enc(f(DF(c \cdot d, g^+, \dot{g})) \star a, msg) \\ &= enc(f(DF(\dot{g}(\{c, d\}), g^+, \dot{g})) \star a, msg) \end{aligned}$$

Here we again select either $c$ or $d$, depending on the definition of $\dot{g}$:

$$enc(f(DF(c, g^+, \dot{g})) \star a, msg) = enc(f(c) \star a, msg)$$

$$enc(f(DF(d, g^+, \dot{g})) \star a, msg) = enc(f(d) \star a, msg)$$

The obtained terms do not contain the addition operation. □

Now we need to show that if a protocol $T_P$ does not contain the operations $+$ and $\cdot$, then there is no difference whether the intruder uses his additional power from $T_{E^\sharp}$ or not. The main idea is to show that if the intruder just stores addends and factors and does not apply $+, \cdot$ to them (this will be modeled by function $DF$), he can still derive $a$ if he could do it before.

First of all, we state an auxiliary lemma that we need to prove in order to get the promised result.

**Lemma 12.** *Let $R \equiv r_1, \ldots, r_k \to r$ be a rule in the theory $T_P \cup T_E$. Let $a_1, \ldots, a_k \to a$ be a ground instance of this rule. Let $(g^+, \dot{g}) \in \overline{\mathbf{Sel}^{+\cdot}}$. Then $DF(a_1, g^+, \dot{g}), \ldots, DF(a_k, g^+, \dot{g}) \to DF(a, g^+, \dot{g})$ is an instance of $R$.*

*Proof:* Let $x_1, \ldots, x_j$ be the variables occurring in the rule $R$. Let $\theta$ be the substitution of $x_1, \ldots, x_j$ with ground terms, such that $r_i\theta \sim^\sharp a_i$ and $r\theta \sim^\sharp a$. Define the substitution $\tilde{\theta}$ by $x_i\tilde{\theta} = DF(x_i\theta, g^+, \dot{g})$. One can easily verify that $r_i\tilde{\theta} = DF(a_i, g^+, \dot{g})$ and $r\tilde{\theta} = DF(a, g^+, \dot{g})$. ■

We can now state a lemma that immediately implies the result we promised to show in this section.

**Lemma 13.** *If $P$ is a protocol that does not contain the operations $+$ and $\cdot$, then for any reduced atom $a$ and any pair of functions $(g^+, \dot{g}) \in \overline{\mathbf{Sel}^{+\cdot}}$:*

$$T_P \cup T_{E^\sharp} \vdash_{\sim^\sharp} a \Longrightarrow T_P \cup T_E \vdash_\sim DF(a, g^+, \dot{g}) \ .$$

*Proof:* The lemma is proved by induction over the derivation length. We need to show how to obtain the derivation *modulo* $\sim$ for each $DF(a, g^+, \dot{g})$.

First of all, note that according to the definition of $DF$ a term that contains neither $+$ nor $\cdot$ will not be modified by $DF$.

Let $\pi = b_1, \ldots, b_l$, be a derivation for $T_P \cup T_{E^\sharp} \vdash_{\sim^\sharp} a$ where $a \sim^\sharp b_l$. The proof is based on induction over the length of $\pi$.

- Base: If $l = 0$, there is no derivation. Since $T_P$ does not contain operations $+$ and $\cdot$ and the rules of $T_{E^\sharp}$ have not been used yet, it means that $a$ contains neither $+$ nor $\cdot$. By definition of $DF$, we have $DF(a, g^+, \dot{g}) = a$.
- Step: Let $\pi_{<l} = b_1, \ldots, b_{l-1}$. By the assumption of the lemma, we know that $a \sim^\sharp b_l$ can be derived from $\pi_{<l}$ by applying a clause from $T_P \cup T_{E^\sharp}$. We need to show that for any $(g^+, \dot{g}) \in \overline{\mathbf{Sel}^{+\cdot}}$, the term $DF(a, g^+, \dot{g})$ can be derived from $\pi_{<l}$ *modulo* $\sim$ using $T_P \cup T_E$.

1) If the rule used to obtain $a$ is in $T_{E^\sharp}$, but not in $T_E$, then it is one of the following cases:

   a) Suppose that $a$ is obtained using the rule

   $$I(x), I(y) \to I(x + y) \ .$$

   In this case, $a = I(t)$ is equivalent to $I(r+s)$ for some terms $I(r)$ and $I(s)$ that have been derived from $\pi_{<l}$. It is possible that the head operation of $r$ or $s$ is $+$. Let $r = r_1 + \ldots + r_{k_r}$ and $s = s_1 + \ldots + s_{k_s}$, where $k_r, k_s \geq 1$ and the head operations of $r_1, \ldots, r_{k_r}, s_1, \ldots, s_{k_s}$ are not $+$.

   By the definition of $DF$, the term $DF(t, g^+, \dot{g})$ is equal to some $DF(r_i, g^+, \dot{g})$ or $DF(s_j, g^+, \dot{g})$, depending on the value of $g^+(\{r_1, \ldots, r_{k_r}, s_1, \ldots, s_{k_s}\})$. Without lessening the generality, assume that for some $i$, the function $g^+$ selects $r_i$ from that set. Define the pair $(\widetilde{g^+}, \widetilde{\dot{g}})$ by initially making them equal to $(g^+, \dot{g})$ and then defining $\widetilde{g^+}(\{r_1, \ldots, r_{k_r}\}) = r_i$. Also change other points of $g^+$ and $\widetilde{\dot{g}}$, such that the conditions put on the pairs of functions in $\overline{\mathbf{Sel}^{+\cdot}}$ continue to hold. It is obvious that $(\widetilde{g^+}, \widetilde{\dot{g}})$ can be defined in this manner. It is also obvious that $DF(r_1 + \cdots + r_{k_r} + s_1 + \cdots + s_{k_s}, g^+, \dot{g}) = DF(r_1 + \cdots + r_{k_r}, \widetilde{g^+}, \widetilde{\dot{g}})$ because at all other points where they are going to be applied, $(g^+, \dot{g})$ and $(\widetilde{g^+}, \widetilde{\dot{g}})$ are equal.

   By induction hypothesis, the atom $I(DF(r, \widetilde{g^+}, \widetilde{\dot{g}}))$ can be derived from $T_P \cup T_E$ *modulo* $\sim$, concluding the induction step.

   b) Suppose that $a$ is obtained using the rule

   $$I(x), I(y) \to I(x \cdot y) \ .$$

   The proof is analogous, only instead of the function $g^+$ we use the function $\dot{g}$.

2) Now suppose $a$ is obtained using a rule $R \equiv r_1, \ldots, r_k \to r$ from $T_P \cup T_E$. In the derivation of $a$, we use the instance of this rule $a_1, \ldots, a_k \to a$, where $a_i$ are atoms derived from $\pi_{<l}$. Let $A := \{a_1, \ldots, a_k\}$. By induction hypothesis, $DF(a_i, g^+, \dot{g})$ for any $a_i \in A$ can be derived from $T_P \cup T_E$. By Lemma 12 we can infer $DF(a, g^+, \dot{g})$ from $DF(A, g^+, \dot{g}) := \{DF(a_i, g^+, \dot{g}) | a_i \in A\}$. ■

Additionally, we need to note that for any atom $a$ that does not contain the operations $+$ and $\cdot$ we have $DF(a, g^+, \dot{g}) = a$. In this way we achieve the promised result.

It is quite obvious that the converse of this lemma does not hold. If the theory $T_P$ contains the atoms $I(h(a))$ and $I(h(b))$ for some operation $h$, and no means to construct other terms of the form $h(\ldots)$, then the atom $I(h(a+b))$ cannot be derived from $T_P \cup T_{E^\sharp}$.

## XI. Conclusions

We have presented an equational theory for bilinear pairings in the symbolic model of cryptography and shown how to reduce Horn theory derivations modulo that equational theory to almost syntactic derivations. We have tested our reduction as a preprocessor for the cryptographic protocol analyzer ProVerif on several pairing-based protocols, affirmed the security of some of them and discovered known attacks for others.

A notable omission in our signature and equational theory is the absence of the treatment of addition (in $G_1$) or multiplication (in $G_T$). While the same omission is also present in existing treatments Diffie-Hellman exponentiation, it is more pronounced in our case, because a sizable number of protocols

(e.g. [26]) make use of it. While the full treatment of addition is most likely intractable, we may try to adapt some ideas of Kremer and Mazaré [7] who allow multiplication in $G_T$, but no addition in $G_1$. A different possible line of future work would be the extension of Mödersheim's results [14] to pairings in order to make verification more efficient.

Our results hold in the symbolic model of cryptography. If one considers a computational semantics of the processes, interpreting $\star$ as the group operation in $G_1$, $\uparrow$ as the group operation in $G_T$ and $e$ as an actual pairing operation from $G_1$ to $G_T$, then one may naturally ask to which extent the secrecy and authenticity properties of the protocol in the symbolic model imply the corresponding properties in the computational model. In the presence of equational theories, it is tricky to relate the properties of symbolic and computational models, even if one considers only passive adversaries [27], [28]. The results of Kremer and Mazaré [7] sidestep these issues, but put many restrictions on the use of the results of pairings (and also consider only the passive adversary). Still, one would expect that at least for authenticity properties, a result mimicking [29] should be possible.

## Acknowledgements

This research was supported by a scholarship from Playtech Estonia, by Estonian Science Foundation, grant #8124, and by the European Regional Development Fund through the Estonian Center of Excellence in Computer Science, EXCS.

## References

[1] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO*, ser. Lecture Notes in Computer Science, J. Kilian, Ed., vol. 2139. Springer, 2001, pp. 213–229.

[2] F. Hess, "Efficient Identity Based Signature Schemes Based on Pairings," in *Selected Areas in Cryptography*, ser. Lecture Notes in Computer Science, K. Nyberg and H. M. Heys, Eds., vol. 2595. Springer, 2002, pp. 310–324.

[3] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 506–522.

[4] R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk, "Universal designated-verifier signatures," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, C.-S. Laih, Ed., vol. 2894. Springer, 2003, pp. 523–542.

[5] A. Joux, "A One Round Protocol for Tripartite Diffie-Hellman," in *ANTS*, ser. Lecture Notes in Computer Science, W. Bosma, Ed., vol. 1838. Springer, 2000, pp. 385–394.

[6] S. S. Al-Riyami and K. G. Paterson, "Tripartite authenticated key agreement protocols from pairings," in *IMA Int. Conf.*, ser. Lecture Notes in Computer Science, K. G. Paterson, Ed., vol. 2898. Springer, 2003, pp. 332–359.

[7] S. Kremer and L. Mazaré, "Computationally sound analysis of protocols using bilinear pairings," *Journal of Computer Security*, vol. 18, no. 6, pp. 999–1033, 2010.

[8] B. Blanchet, "An efficient cryptographic protocol verifier based on prolog rules," in *CSFW*. IEEE Computer Society, 2001, pp. 82–96.

[9] R. Küsters and T. Truderung, "Using ProVerif to Analyze Protocols with Diffie-Hellman Exponentiation," in *CSF*. IEEE Computer Society, 2009, pp. 157–171.

[10] D. Dolev and A. C.-C. Yao, "On the Security of Public Key Protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–207, 1983.

[11] R. Küsters and T. Truderung, "Reducing protocol analysis with XOR to the XOR-free case in the horn theory based approach," in *ACM Conference on Computer and Communications Security*, P. Ning, P. F. Syverson, and S. Jha, Eds. ACM, 2008, pp. 129–138.

[12] M. Abadi, B. Blanchet, and C. Fournet, "Just fast keying in the pi calculus," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 3, 2007.

[13] B. Blanchet, M. Abadi, and C. Fournet, "Automated verification of selected equivalences for security protocols," *J. Log. Algebr. Program.*, vol. 75, no. 1, pp. 3–51, 2008.

[14] S. Mödersheim, "Diffie-Hellman without Difficulty," in *Proceedings of the 8th International Workshop on Formal Aspects of Security & Trust (FAST2011)*, 2011.

[15] B. Blanchet and A. Podelski, "Verification of cryptographic protocols: Tagging enforces termination," in *FoSSaCS*, ser. Lecture Notes in Computer Science, A. D. Gordon, Ed., vol. 2620. Springer, 2003, pp. 136–152.

[16] M. Abadi and P. Rogaway, "Reconciling two views of cryptography (the computational soundness of formal encryption)," *J. Cryptology*, vol. 15, no. 2, pp. 103–127, 2002.

[17] M. Abadi and V. Cortier, "Deciding knowledge in security protocols under (many more) equational theories," in *CSFW*. IEEE Computer Society, 2005, pp. 62–76.

[18] S. Ciobâca, S. Delaune, and S. Kremer, "Computing knowledge in security protocols under convergent equational theories," in *CADE*, ser. Lecture Notes in Computer Science, R. A. Schmidt, Ed., vol. 5663. Springer, 2009, pp. 355–370.

[19] S. Goldwasser and S. Micali, "Probabilistic Encryption," *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270–299, Apr. 1984.

[20] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," in *CRYPTO*, ser. Lecture Notes in Computer Science, M. Yung, Ed., vol. 2442. Springer, 2002, pp. 354–368.

[21] A. Pankova and P. Laud, "Symbolic Analysis of Cryptographic Protocols Containing Bilinear Pairings," Cybernetica AS, Tech. Rep. T-4-16, 2012.

[22] Z. Cheng, L. Vasiu, and R. Comley, "Pairing-based one-round tripartite key agreement protocols," Cryptology ePrint Archive, Report 2004/079, 2004.

[23] K. Shim, "Efficient one round tripartite authenticated key agreement protocol from Weil pairing," *Electronics Letters*, vol. 39, no. 2, pp. 208–209, 2003.

[24] K.-A. Shim and S. S. Woo, "Cryptanalysis of tripartite and multi-party authenticated key agreement protocols," *Inf. Sci.*, vol. 177, no. 4, pp. 1143–1151, 2007.

[25] S. Ciobâca and V. Cortier, "Protocol Composition for Arbitrary Primitives," in *CSF*. IEEE Computer Society, 2010, pp. 322–336.

[26] N. Smart, "Identity-based authenticated key agreement protocol based on Weil pairing," *Electronics Letters*, vol. 38, no. 13, pp. 630–632, 2002.

[27] M. Baudet, V. Cortier, and S. Kremer, "Computationally sound implementations of equational theories against passive adversaries," *Inf. Comput.*, vol. 207, no. 4, pp. 496–520, 2009.

[28] G. Bana, P. Mohassel, and T. Stegers, "Computational soundness of formal indistinguishability and static equivalence," in *ASIAN*, ser. Lecture Notes in Computer Science, M. Okada and I. Satoh, Eds., vol. 4435. Springer, 2006, pp. 182–196.

[29] V. Cortier and B. Warinschi, "Computationally sound, automated proofs for security protocols," in *ESOP*, ser. Lecture Notes in Computer Science, S. Sagiv, Ed., vol. 3444. Springer, 2005, pp. 157–171.