Symbolic Analysis of Cryptographic Protocols Containing Bilinear Pairings

Alisa Pankova, Peeter Laud
Copyright ©2012
Alisa Pankova¹, Peeter Laud².
¹ University of Tartu, Institute of Computer Science,
² Cybernetica, Institute of Information Security

The research reported here was supported by:

1. Scholarship from Playtech Estonia

2. the European Regional Development Fund through the Estonian Center of Excellence in Computer Science, EXCS

3. Estonian Science foundation, grant(s) No. 8124

All rights reserved. The reproduction of all or part of this work is permitted for educational or research use on condition that this copyright notice is included in any copy.

Cybernetica research reports are available online at
http://research.cyber.ee/

Mailing address:
AS Cybernetica
Akadeemia tee 21
12618 Tallinn
Estonia
Symbolic Analysis of Cryptographic Protocols Containing Bilinear Pairings

Alisa Pankova, Peeter Laud

June 20, 2012

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.

1 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 [BF01, Hes02], encryption systems with keyword search [BCOP04], designated verifier signatures [SBWP03], efficient key-agreement protocols [Jou00, ARP03], 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 [KM10], 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 [Bla01], one of the most successful tools for cryptographic protocol verification. By extending the theory transformation
methods of Küsters and Truderung [KT09], we can change the problem of derivation \textit{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 [KT09].
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. 2 and Horn theory
based protocol analysis in Sec. 3, we describe our symbolic model for bilinear pairings in Sec. 4.
Sec. 5–8 deal with making the inference \textit{modulo} properties of bilinear pairings tractable for ProVerif.
Sec. 6 extends the notion of \textit{exponent ground} theories [KT09] to pairings. Sec. 5 presents the ideas
of theory transformation at term level, while Sec. 7 and 8 show the actual transformation and prove
its results sound (Sec. 8) and complete (Sec. 7) with respect to our theory of bilinear pairings.

We finish the paper with an overview of our experimental results in Sec. 9, with the analysis
of the adversarial power in the presence of group operations (Sec. 10) and with the conclusion in
Sec. 11.

2 Related work

We perform the analysis of protocols in the \textit{symbolic cryptography} [DY83] model, using the ProVerif
protocol analyzer [Bla01]. Our reduction from Horn theories \textit{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 [KT08] and Diffie-Hellman
exponentiation [KT09] 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 [ABF07, BAF08]). Recently, Mödersheim [Möd11] has shown
that in certain well-tagged [BP03] cases, ProVerif’s treatment may be equivalent to Küsters and
Truderung [KT09].

By our knowledge, bilinear pairings in symbolic cryptography have been considered only by
Mazaré and Kremer [KM10]. 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 [AR02].

The task considered in this paper is an instance of the general problem of inference in Horn the-
ories \textit{modulo} equational theories. While the static equivalence of terms is usually decidable [AC05,
CDK09], it is much harder to perform an actual protocol analysis \textit{modulo} an equational theory
[BAF08].

3 Modeling Protocols with Horn Theories

Let \( \Sigma \) be a finite signature. A \textit{term} \( t \) over \( \Sigma \) is either a variable \( x \) from a countable set \( V \), a name
\( m \) from a countable set \( N \) or \( f(t_1, \ldots, t_{\text{var}(f)}) \) for \( f \in \Sigma \). By \( \text{var}(t) \) we denote the set of variables
occurring in term \( t \). A term is \textit{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 \( \text{enc}(\cdot, \cdot) \) for modeling deterministic
encryption and \textit{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 \textit{atom}. It is \textit{ground} if \( t_1, \ldots, t_k \) are ground. A \textit{Horn clause} has the form \( a_1, \ldots, a_n \rightarrow a_0 \), where \( a_0, \ldots, a_n \) are atoms. A \textit{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 \( b_i = a \) and for each \( j \in \{1, \ldots, l\} \), \( b_j \) is a ground atom and there exists a clause \( a_1, \ldots, a_n \rightarrow a_0 \in T \) 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 \textit{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 \textit{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 [Bla01] 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 \( 1, \ldots, i \) of a protocol session, it can also obtain the message \( i+1 \) (presumably by handing some previous message over to a participant). This model introduces certain abstractions (e.g. one cannot naturally state that messages 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 \( n \)?

4 Equational Theory for Bilinear Pairings

In the computational model of cryptography [GM84], a (symmetric) \textit{bilinear} pairing [Jou00, BF01] is an efficiently computable non-degenerate mapping \( e : G_1 \times G_1 \rightarrow G_T \), 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 [BF01, BKLS02] 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 \( aPbP \in G_1 \) 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 \( \sim^{-1} \) (for modeling negation in \( G_1 \), inverse in \( G_T \), and inverses in the exponents), and the binary operation \( e \) (pairing). Similarly to [KT09] (and all other approaches so far), we omit additions in \( G_1 \) / multiplications in \( G_T \) from our signature (see Sec. 10 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.

\[
\begin{align*}
(x \uparrow y) \uparrow z \sim (x \uparrow z) \uparrow y \\
e(x, y) \sim e(y, x) \\
(x \uparrow y) \uparrow y^{-1} \sim x \\
(x \star y) \star y^{-1} \sim x \\
x^{-1} \star y \sim (x \star y)^{-1} \\
e(x, y \star z) \sim e(x, y) \uparrow z
\end{align*}
\]

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

\[
\begin{align*}
I(x), I(y) \rightarrow I(e(x, y)) \\
I(x), I(y) \rightarrow I(x \star y) \\
I(x), I(y) \rightarrow I(x \uparrow y) \\
I(x) \rightarrow I(x^{-1})
\end{align*}
\]

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 \( T_P \cup T_E \not\vdash \sim I(m) \) means that the intruder cannot get the message \( m \) even when employing the algebraic properties of bilinear mappings.

## 5 Exponent-ground Theory for Bilinear Pairings

Küsters and Truderung [KT09], constrain the theory for Diffie-Hellman exponentiation \( T_{DH} \) to a theory \( T_{C \, DH} \) 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 \( t \uparrow s^{-1} \) 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 \(*\) 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. \(I(x), I(y) \rightarrow I(e(x,y))\);
2. \(I(x), I(e) \rightarrow I(x \uparrow e)\) for each \(e \in C\);
3. \(I(x), I(e) \rightarrow I(x \uparrow e^{-1})\) for each \(e \in C\);
4. \(I(x), I(e) \rightarrow I(x \star e)\) for each \(e \in C\);
5. \(I(x), I(e) \rightarrow I(x \star e^{-1})\) for each \(e \in C\).

Here "for each \(e \in C\)" 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 \(C^{-1} = \{e^{-1} | e \in C\}\), and let \(C^* = C \cup C^{-1}\). The function \(\delta_C\) is defined inductively:

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

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. $\delta_C(\delta_C(t)) = \delta_C(t)$.

**Proof:** This lemma summarizes the properties of the function $\delta_C$. The proof of first two points is based on induction and case distinction. We have to look through all the possible cases of application of $\delta_C$. The third point is directly implied by the first two.

1. $\delta_C(t)$ is $C$-exponent ground.
   - $t = x$ for a variable $x$: $\delta_C(x) = x$, and a variable is considered to be $C$-exponent-ground since it does not use any exponents at all.
   - $t = t'^{-1}$ for a term $t'$: $\delta_C(t'^{-1}) = \delta_C(t')$. The term $\delta_C(t')$ is $C$-exponent-ground according to induction hypothesis.
   - $t = t' \uparrow s$, where $t'$ is some term and $s \in C^*$. $\delta_C(t' \uparrow s) = \delta_C(t') \uparrow s$. Since $t'$ is $C$-exponent-ground by induction hypothesis, and $s \in C^*$, the whole term is also $C$-exponent-ground.
   - $t = t' \uparrow s$, where $t'$ is some term and $s \notin C^*$. $\delta_C(t' \uparrow s) = \delta_C(t')$, and $\delta_C(t')$ is $C$-exponent-ground according to induction hypothesis.
   - $t = t' \star s$: the proof is analogical to exponentiation cases.
   - $t = f(t_1, \ldots, t_n)$ for some terms $t_1, \ldots, t_n$.
     $\delta_C(f(t_1, \ldots, t_n)) = f(\delta_C(t_1), \ldots, \delta_C(t_n))$. The arguments of $f$, namely $\delta_C(t_1), \ldots, \delta_C(t_n)$, are $C$-exponent-ground by induction hypothesis, and after applying the function $f$, the term is still $C$-exponent-ground.

2. $\delta_C(t) = t$ iff $t$ is $C$-exponent ground.
   The function $\delta_C$ transforms arbitrary terms to $C$-exponent-ground terms. Hence if we take $t$ that is not $C$-exponent-ground, the term $\delta_C(t)$ still must be $C$-exponent-ground, and therefore $\delta_C(t) \neq t$. It remains to prove that if $t$ is $C$-exponent-ground, then $\delta_C(t) = t$.
   - $t = x$ for a variable $x$. A variable is $C$-exponent-ground. $\delta_C(x) = x = t$.
   - $t = t'^{-1}$ for a term $t'$. The term $t'$ is not $C$-exponent-ground, so we may omit this case.
   - $t = t' \uparrow s$, where $t'$ is some term and $s \in C^*$. $\delta_C(t' \uparrow s) = \delta_C(t') \uparrow s$. If $t'$ is $C$-exponent-ground, it means that $t'$ should also be $C$-exponent-ground (otherwise $t$ would not be). By induction hypothesis, $t' = \delta_C(t')$. We get that $\delta_C(t) = t' \uparrow s$.
   - $t = t' \uparrow s$, where $t'$ is some term and $s \notin C^*$. The term $t$ is not $C$-exponent-ground.
   - $t = t' \star s$: the proof is analogical to exponentiation.
   - $t = f(t_1, \ldots, t_n)$ for some terms $t_1, \ldots, t_n$.
     $\delta_C(f(t_1, \ldots, t_n)) = f(\delta_C(t_1), \ldots, \delta_C(t_n))$. The arguments of $f$, namely $\delta_C(t_1), \ldots, \delta_C(t_n)$, should be $C$-exponent-ground (otherwise $t$ would not be), and, by induction hypothesis, $f(\delta_C(t_1), \ldots, \delta_C(t_n)) = f(t_1, \ldots, t_n) = t$.

3. $\delta_C(\delta_C(t)) = \delta_C(t)$.
   Since from the two previous points we know that $\delta_C(t)$ is $C$-exponent ground and that $\delta_C$ does not modify the terms that are already $C$-exponent ground, we get $\delta_C(\delta_C(t)) = \delta_C(t)$.
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 $\delta_C(t) \sim \delta_C(s)$.

**Proof:** We will first consider the case where $s$ is a reduced form of $t$. In this case, the proof is done by induction over the structure of $t$:

1. If the head operation of $t$ is neither $\ast, \uparrow$ nor $\uparrow$, then the term $s$ should have the same head operation as $t$ has, since the reduction does not modify any other functional symbols. There are now two cases:
   
   (a) If $t$ is a constant or a variable, then $\delta_C(t) = t = s = \delta_C(s)$.
   
   (b) If $t$ is of the form $f(t_1, \ldots, t_n)$, then $s$ should be of the form $f(s_1, \ldots, s_n)$, where $s_i$ is a reduced form of $t_i$. By the induction hypothesis, $\delta_C(t_i) \sim \delta_C(s_i)$, and therefore $\delta_C(t) = f(\delta_C(t_1), \ldots, \delta_C(t_n)) \sim f(\delta_C(s_1), \ldots, \delta_C(s_n)) = \delta_C(s)$.

2. If the head operation of $t$ is $\uparrow$, then $t = u^{-1}$ for some $u$. There are now two cases for $u$:
   
   (a) If $u$ is of the form $r^{-1}$ for some $r$, then $s$ is a reduced form of $r$ and, based on induction hypothesis, $\delta_C(t) = \delta_C(r) \sim \delta_C(s)$.
   
   (b) If $u$ is not of the form $r^{-1}$ for some $r$, then $s$ should be of the form $w^{-1}$ where $w$ is a reduced form of $u$. By the induction hypothesis, $\delta_C(u) \sim \delta_C(w)$, and since $\delta_C(t) = \delta_C(u)$ and $\delta_C(s) = \delta_C(w)$ (from the definition of $\delta_C$), we have $\delta_C(t) \sim \delta_C(s)$.

3. If the head operation of $t$ is $\uparrow$, we may take into account all the sequential $\uparrow$ head operations of $t$ and write $t$ as $t_0 \uparrow t_1 \uparrow \ldots \uparrow t_n$, where the head symbol of $t_0$ is not $\uparrow$.
   
   (a) There are $i, j \in \{1, \ldots, n\}$ such that $t_i \sim t_j^{-1}$. These two terms in fact cancel each other out, and if we remove these terms from $t$ and obtain $t'$, we get that $s$ is a reduced form of $t'$. Because $t$ is exponent-reduced, it follows that $t_i = t_j^{-1}$ or $t_j = t_i$, and we get that $t_i \in C^*$ iff $t_j \in C^*$. We have $\delta_C(t) \sim \delta_C(t')$, and by induction hypothesis $\delta_C(t') \sim \delta_C(s)$, which implies $\delta_C(t) \sim \delta_C(s)$.
   
   (b) If there are no such $i$ and $j$, then $s$ must be of the form $s_0 \uparrow s_1 \uparrow \ldots \uparrow s_n$, where head symbol of $s_0$ is not $\uparrow$, $s_0$ is a reduced form of $t_0$, and $t_i = s_i$ for each $i \in \{1, \ldots, n\}$ ($t$ is exponent-reduced implies that all $t_i$-s are exponent-reduced). Now, $\delta_C(t) = \delta_C(t_0) \uparrow t_1 \uparrow \ldots \uparrow t_n$, where $t_1, \ldots, t_n$ are exactly these elements of $t_1, \ldots, t_n$ which belong to $C^*$. We have that $t_i \in C^*$ iff $s_i \in C^*$ for each $i \in \{1, \ldots, n\}$, and therefore $\delta_C(s) = \delta_C(s_0) \uparrow s_1 \uparrow \ldots \uparrow s_n$. By the induction hypothesis, $\delta_C(t_0) = \delta_C(s_0)$. It follows that $\delta_C(t) \sim \delta_C(s)$.

4. If the head symbol of $t$ is $\ast$, we may write $t$ as $t_0 \ast t_1 \ast \ldots \ast t_n$, where the head symbol of $t_0$ is not $\ast$. We have to look through exactly the same cases as in the exponentiation.

Consider now arbitrary $t$ and $s$, such that $t \sim s$. Let $r$ be a reduced form of $t$. In this case, $r$ is also a reduced form of $s$. By the argument above, $\delta_C(t) \sim \delta_C(r) \sim \delta_C(s)$ and the transitivity of $\sim$ implies $\delta_C(t) \sim \delta_C(s)$.
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.

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

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 instantiated by some substitution. 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 $t$.

Proof: The proof is done by induction over the structure of $t$:

1. If $t$ is a standard term, $\delta_C(t\theta) = t\delta_C(\theta)$ since $\delta_C$ does not modify $t$ at all. Let it be the induction basis.
2. If $t = s^{-1}$ for some $s$, then $t$ is not a $C$-exponent-ground term. We do not have to consider this case.
3. If $t = s \uparrow s'$, it follows that $s$ is $C$-exponent-ground and $s' \in C^\star$. We have $\delta_C(t\theta) = \delta_C(s\theta \uparrow s') = \delta_C(s\theta) \uparrow s'$. By the induction hypothesis, $\delta_C(s\theta) = s\delta_C(\theta)$. Thus, $\delta_C(s\theta \uparrow s') = s\delta_C(\theta \uparrow s') = t\delta_C(\theta)$.
4. If $t = s \star s'$, the proof is analogous to exponentiation.
5. If $t = e(s, s')$ for some $C$-exponent-ground terms $s$ and $s'$, then $e$ can be considered as an ordinary functional symbol since $\delta_C$ has no special definition for $e$. $\delta_C(t\theta) = \delta_C(e(s\theta, s'\theta)) = e(\delta_C(s\theta), \delta_C(s'\theta)) = e(s\delta_C(\theta), s'\delta_C(\theta)) = e(s, s')\delta_C(\theta) = t\delta_C(\theta)$

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.

$$
\delta_C(t\theta) = \delta_C((e(x, y) \uparrow c_1)\theta) = \delta_C(e(u^{-1}, v \star w) \uparrow c_1) = \delta_C(e(u^{-1}, v \star w) \uparrow c_1) = e(\delta_C(u^{-1}), \delta_C(v \star w)) \uparrow c_1 = e(u, \delta_C(v \star w)) \uparrow c_1 = e(u, \delta_C(v \star w)) \uparrow c_1 = e(u, \delta_C(v \star w)) \uparrow c_1
$$
\[ t\delta_C(\theta) = t\delta_C(u^{-1}/x, v \ast w/y) = t(\delta_C(u^{-1})/x, \delta_C(v \ast w)/y) = t(u/x, v/y) = \epsilon(x, y) \uparrow c_1(u/x, v/y) = \epsilon(u, v) \uparrow c_1 \]

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. \( \square \)

**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 \( T_E \) instead of the rules from \( T_E \).

**Proof:** Let \( \pi = b_1, \ldots, b_l \) be a derivation for \( T \cup T_E \vdash a \), where \( b_1 \sim a \). We can assume that the \( a \), the substitution \( \theta \), and therefore all \( b_i \)'s are reduced. When constructing a protocol, we have to reduce its terms first.

We need to show that \( \delta_C(\pi) \) is a derivation for \( T \cup T_E \vdash a \). This completes the proof, because \( \delta_C(\pi) \) is C-exponent-ground by Lemma 2.

Because \( b_1 \sim a \) and both \( b_l \) and \( a \) are reduced, by Lemma 3, we have \( \delta_C(b_1) \sim \delta_C(a) \). By Lemma 2, \( \delta_C(a) = a \) since \( a \) is C-exponent-ground, so we have \( \delta_C(b_1) \sim a \). To prove that \( \delta_C(\pi) \) is a derivation for \( T \cup T_E \vdash a \), we only need to show for each \( i \in \{1, \ldots, l\} \) that \( \delta_C(b_i) \) can be obtained from \( \{\delta_C(b_1), \ldots, \delta_C(b_{i-1})\} \) by applying one of the Horn clauses from \( T \cup T_E \). We need to consider five cases: whether \( b_i \) is obtained by one of the clauses in theory \( T \) or one of the rules defined in the theory \( T_E \).

Let us look through all these cases:

1. \( b_i \) is obtained by applying some C-exponent-ground clause from \( T \). There exists a clause \( a_1, \ldots, a_n \rightarrow a_0 \) in \( T \) such that \( a_0, \ldots, a_n \) are C-exponent-ground. There exists a substitution \( \theta \) such that \( a_0\theta \sim b_i \) and for each \( j \in \{1, \ldots, n\} \) there exists \( k_j \in \{1, \ldots, n\} \) with \( a_j\theta \sim b_{k_j} \). Since \( a_j \) is C-exponent-ground and \( \theta \) is reduced, \( a_j\theta \) is exponent-reduced. By Lemma 3, \( \delta_C(a_j\theta) \sim \delta_C(b_{k_j}) \) for all \( j \in \{0, \ldots, n\} \). By Lemma 4, \( a_j\delta_C(\theta) \sim \delta_C(b_{k_j}) \). We can apply the same clause \( a_1, \ldots, a_n \rightarrow a_0 \) in \( T \) with the substitution \( \delta_C(\theta) \) to \( \delta_C(b_{k_1}), \ldots, \delta_C(b_{k_n}) \) and obtain \( \delta_C(b_{h_n}) = \delta_C(b_k) \).

2. \( b_i \) is obtained by applying \( I(x), I(y) \rightarrow I(\epsilon(x, y)) \). In this case, \( b_i \) is of the form \( I(t) \), and, for some \( j, k < i \), the atom \( b_j \) is of the form \( I(s) \), and the atom \( b_k \) is of the form \( I(r) \) such that \( t \sim \epsilon(s, r) \). We need to show that \( \delta_C(I(t)) \) can be obtained from \( I(\delta_C(s)) \) and \( I(\delta_C(r)) \). Since \( s \) and \( r \) are reduced by assumption, by Lemma 3 we have \( \delta_C(I(t)) \sim \epsilon(\delta_C(s), \delta_C(r)) \). We apply the clause \( I(x), I(y) \rightarrow I(\epsilon(x, y)) \) from theory \( T_E \) to get \( I(\delta_C(\epsilon(s, r))) \) from \( I(\delta_C(s)) \) and \( I(\delta_C(r)) \).
3. $b_i$ is obtained by applying $I(x) \rightarrow I(x^{-1})$. In this case, $b_i$ is of the form $I(t)$ and, for some $j < i$, the atom $b_j$ is of the form $I(s)$ with $t \sim s^{-1}$. Since $t$ and $s$ are reduced and thus both $t$ and $s^{-1}$ are exponent-reduced, we use Lemma 3 to obtain $\delta_C(t) \sim \delta_C(s^{-1}) = \delta_C(s)$. Hence, $\delta_C(b_i) = I(\delta_C(t)) \sim I(\delta_C(s)) = \delta_C(b_j)$.

4. $b_i$ is obtained by applying $I(x), I(y) \rightarrow I(x \uparrow y)$. In this case, $b_i$ is of the form $I(t)$ and there are atoms $I(s)$ and $I(r)$ amongst $b_1, \ldots, b_{i-1}$ such that $t \sim s \uparrow r$. We need to show that $I(\delta_C(t))$ can be obtained from $I(\delta_C(s))$ and $I(\delta_C(r))$.

Since $s$ and $r$ are reduced, $s \uparrow r$ is exponent-reduced. By Lemma 3, we have $\delta_C(t) = \delta_C(s \uparrow r)$, so it is enough to show that $I(\delta_C(s \uparrow r))$ can be obtained from $I(\delta_C(s))$ and $I(\delta_C(r))$. Consider three subcases:

(a) If $r \notin C^*$, then $\delta_C(s \uparrow r) = \delta_C(s)$.

(b) If $r \in C$, then $\delta_C(r) = r$, and therefore $\delta_C(s \uparrow r) = \delta_C(s) \uparrow r = \delta_C(s) \uparrow \delta_C(r)$. $I(\delta_C(s \uparrow r))$ can be obtained from $I(\delta_C(s))$ and $I(\delta_C(r))$ using the rule $I(x), I(c) \rightarrow I(x \uparrow c)$.

(c) If $r \in C^1$, then $r = \delta_C(r)^{-1}$ and therefore $\delta_C(s \uparrow t) = \delta_C(s) \uparrow t = \delta_C(s) \uparrow \delta_C(r)^{-1}$. $I(\delta_C(s \uparrow r))$ can be obtained from $I(\delta_C(s))$ and $I(\delta_C(r))$ using the rule $I(x), I(c) \rightarrow I(x \uparrow c^{-1})$.

5. $b_i$ is obtained by applying $I(x), I(y) \rightarrow I(x \ast y)$. In this case, $b_i$ is of the form $I(t)$ and there are atoms $I(s)$ and $I(r)$ amongst $b_1, \ldots, b_{i-1}$ such that $t \sim s \ast r$. We need to show that $I(\delta_C(t))$ can be obtained from $I(\delta_C(s))$ and $I(\delta_C(r))$, and it can be done in the same way as it was done for exponentiation, using the rules $I(x), I(c) \rightarrow I(x \ast c)$ and $I(x), I(c) \rightarrow I(x \ast c^{-1})$.

6 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 [KT09], 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 $C = \{c_1, \ldots, c_n\}$ be the set of pure, ground terms used in the derivation according to the theory $T$ using the signature $\Sigma$.

Define $\Sigma_{pair} = (\Sigma \setminus \{\uparrow, \downarrow, \ast\}) \cup \{0, s, p, exp, mult\}$ as the new signature.

The constant 0 and the unary functions $s$ and $p$ are used for encoding integers, as in [KT09]. The integer $n$ will be encoded as $s^n(0) = s \ldots s(0) \ldots$, and $-n$ as $p^n(0)$. 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 $s \ast c_{1}^{n_1} \ast \ldots \ast c_{m}^{n_m}$ 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 $exp(e(x, y), s^t(0), 0, p(0))$.

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

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

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

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

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

$$
\text{incr}^X_i(t) = \text{incr}^X_i(X(t, 0, \ldots, 0)) \quad \text{and} \quad \text{decr}^X_i(t) = \text{decr}^X_i(X(t, 0, \ldots, 0)).
$$

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

- $\text{incr}^\exp_i(\text{exp}(e(x, y), 0, p(0), 0)) = \text{exp}(e(x, y), 0, 0, 0)$;
- $\text{incr}^\text{mult}_i(x) = \text{mult}(x, s(0), 0, 0)$;
- $\text{decr}^\exp_i(\text{exp}(y, s(s(0)), 0, 0)) = \text{exp}(y, s(0), 0, 0)$.  

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

- $\langle x \rangle = x$ for a variable or name $x$;
- $\langle f(t_1, \ldots, t_n) \rangle = f(\langle t_1 \rangle, \ldots, \langle t_n \rangle)$ (if $\not\in \{\uparrow, \downarrow, \ast, \circ\}$);
- $\langle t \uparrow c_i \rangle = \text{incr}^\exp_i(\langle t \rangle)$;
- $\langle t \downarrow c_i \rangle = \text{decr}^\exp_i(\langle t \rangle)$;
- $\langle t \downarrow c_i \rangle = \text{incr}^\text{mult}_i(\langle t \rangle)$;
- $\langle t \downarrow c_i \rangle = \text{decr}^\text{mult}_i(\langle t \rangle)$;
- $\langle e(t_1 \ast c_1, t_2) \rangle = \langle e(t_1, t_2) \uparrow c_i \rangle$;
- $\langle e(t_1, t_2 \ast c_1) \rangle = \langle e(t_1, t_2) \uparrow c_i \rangle$;
- $\langle e(t_1, t_2) \rangle = \langle e(\langle t_1 \rangle, \langle t_2 \rangle) \rangle$ (only if the two previous rules do not apply);
- $\langle p(t) \rangle = p(\langle t \rangle)$, for an atom $p(t)$.

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

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

**Proof:** The proof of this lemma is similar to the proof of [KT09]. It becomes more complex since there are additional definitions regarding the terms whose head symbol is pairing function $e$.

Let $t$ and $s$ be $C$-exponent ground terms. Assume that $t \sim s$. There exists a term $r$ which is a reduced form of both $t$ and $s$. We can obtain $r$ from $s$ or from $t$ applying the equations defined for bilinear mappings:
1. \((x \uparrow y) \uparrow z = (x \uparrow z) \uparrow y; (x \star y) \star z = (x \star z) \star y\) can be applied from left to right and from right to left a number of times. The transformation preserves \(\rceil \cdot \lceil\) according to the definition of \(\rceil \cdot \lceil\). Let \(y = c_i\) and \(z = c_j\). Let \(x\) be some \(C\)-exponent ground term.

- \(\rceil (x \uparrow c_i) \uparrow c_j \lceil = incr_i^{exp}(\rceil x \uparrow c_i \lceil) = incr_i^{exp}(incr_j^{exp}(\rceil x \lceil))\).
- \(\rceil (x \uparrow c_j) \uparrow c_i \lceil = incr_j^{exp}(\rceil x \uparrow c_j \lceil) = incr_j^{exp}(incr_i^{exp}(\rceil x \lceil))\).

Here we need to show that the metatheoretical functions of the form \(incr_i^{exp}\) and \(decr_i^{exp}\) commute. This property is obvious from the metatheoretical meaning of these functions. It is not hard to verify it for example by trying to reorder all the possibilities from the definition of \(incr_i^{exp}\) and \(decr_i^{exp}\). There are too many different cases, and they are not brought here (this proof is also not shown in details in [KT09]).

The proof for \(y = c_i^{-1}, z = c_j^{-1}\), and \(\star\) operation is analogical.

2. \(e(x, y \star z) = e(x, y) \uparrow z\) can be applied from left to right and from right to left a number of times. Let \(z = c_i\), and let \(x, y\) be some \(C\)-exponent-ground terms. We get that \(\rceil e(x, y \star c_i) \lceil = \rceil e(x, y) \uparrow c_i \lceil\) directly from the definition of \(\rceil \cdot \lceil\).

3. \((x \uparrow y) \uparrow y^{-1} = x; (x \star y) \star y^{-1} = x\) can be applied from left to right a number of times. The transformation preserves \(\rceil \cdot \lceil\) according to the definition of \(\rceil \cdot \lceil\). Let \(y = c_i\), we get \(\rceil (x \uparrow y) \uparrow y^{-1} \lceil = decr_i^{exp}(\rceil x \uparrow y \lceil) = incr_i^{exp}(decr_i^{exp}(\rceil x \lceil)) = \rceil x \lceil\).

4. \((x^{-1})^{-1} = x; e(x, y^{-1}) = e(x, y)^{-1}\) cannot be applied, since \(t\) and \(s\) are \(C\)-exponent-ground and all the transformations preserve \(C\)-exponent-groundness.

5. \(e(x, y) = e(y, x)\). There are two subcases for this formula.

- Suppose that the head symbol of one of the terms \(x\) and \(y\) is \(*\). Let \(x = z \star c_i\) for some \(C\)-exponent-ground term \(z\). We get:
  - \(\rceil e(z \star c_i, y) \lceil = \rceil e(z, y) \star c_i \lceil\).
  - \(\rceil e(y, z \star c_i) \lceil = \rceil e(y, z) \star c_i \lceil\).

Here \(e(z, y) \sim e(y, z)\), and therefore \(e(z, y) \star c_i \sim e(z, y) \star c_i\) which by induction hypothesis implies \(\rceil e(z, y) \star c_i \lceil = \rceil e(z, y) \star c_i \lceil\).

The case for \(y = z \star c_i\) is analogical. It is important that the rule \(\rceil e(x, y) \lceil = e(\rceil x \lceil, \rceil y \lceil)\) will not be applied until all the \(\star\) symbols will be handled.

- If the head symbol of \(x\) and \(y\) is not \(*\), we get \(e(\rceil x \lceil, \rceil y \lceil) \sim e(\rceil y \lceil, \rceil x \lceil)\) since \(x \sim \rceil x \lceil\) and \(y \sim \rceil y \lceil\) according to induction hypothesis.

It is important that we do not achieve syntactical equality just by applying \(\rceil \cdot \lceil\), and this property is not encoded. In ProVerif, we will define this equation in the heading if it is necessary for the protocol. In this theory, we assume that the equation \(e(x, y) = e(y, x)\) holds also for the \(C\)-exponent-ground theory \(T_C\), and it does not have to be proved by encoding.

Example 5. Let \(C = \{c_i\}\). Let \(t = enc(e(g \star c_i, g \star c_i^{-1}), m)\) where \(g\) and \(m\) are some variables.
\( \forall t \forall n \quad \vdash e(g \ast c_1, g \ast c_i^{-1}, m) \gamma \\
= \quad e(e(g \ast c_1, g \ast c_i^{-1}) \gamma, \forall m \gamma) \\
= \quad e(e(g, g \ast c_i^{-1}) \uparrow c_i^{-1}, m) \\
= \quad e(incr^{-1}_1(e(g, g \ast c_i^{-1}) \gamma), m) \\
= \quad e(incr^{-1}_1(e(g, g) \uparrow c_i^{-1}), m) \\
= \quad e(decr^{-1}_1(e(g, g)) \gamma, m) \\
= \quad e(decr^{-1}_1(e(g, g) \uparrow g)), m) \\
= \quad e(e(g, g), m) \\
\square

7 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 \( \vdash_{\sim} \) according to \( T_C \). Formally, the derivation \( \vdash_{\sim} \) 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 [KT09], 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 \( exp(x, x_1, \ldots, x_m) \), the intruder can produce \( exp(x, x_1, \ldots, s(x_1), \ldots, x_m) \) 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 [KT09]. Namely, we must handle the addition of exponents. The term \( e(x \ast c(x_1) \ast \ldots \ast c(x_m), y \ast c(y_1) \ast \ldots \ast c(y_m)) \) is equivalent to \( e(x, y) \uparrow c(x_1) \uparrow \ldots \uparrow c(x_m) \), where for each \( i \): \( z_i = x_i + y_i \); these terms have the same encoding. We handle the addition by introducing a predicate \( A \). Metatheoretically, \( A(x, y, z) \) is true iff \( z = t2i(t2i(x) + t2i(y)) \).

The predicate \( A \) cannot be defined through case enumeration. We define it recursively, using auxiliary predicates \( INC \) 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 [KT09] 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 \( x \ast y \sim z \).
- \( P(x, y, z) \) 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
Figure 1: Generic clauses of the theory $T$: 

1. $\forall c, t \in C, m \in \mathbb{N} \ (\text{mult}(c, t, m, 0) \rightarrow \text{mult}(c, t, m, 0))$ for each $c \in C$ and $t \in \mathbb{N}$.
2. $\forall p \in \mathbb{N} \ (\text{power}(p, 0, p) \rightarrow \text{power}(p, 0, p))$.
3. $\forall c, t \in C, m \in \mathbb{N} \ (\text{power}(c, t, m, 0) \rightarrow \text{power}(c, t, m, 0))$ for each $c \in C$ and $t \in \mathbb{N}$.
4. $\forall c, t \in C, m, n \in \mathbb{N} \ (\text{power}(c, t, m, n) \rightarrow \text{power}(c, t, m, n))$ for each $c \in C$ and $t \in \mathbb{N}$.
5. $\forall c, t \in C, m \in \mathbb{N} \ (\text{power}(c, t, m, 0) \rightarrow \text{power}(c, t, m, 0))$ for each $c \in C$ and $t \in \mathbb{N}$.
6. $\forall c, t \in C, m, n \in \mathbb{N} \ (\text{power}(c, t, m, n) \rightarrow \text{power}(c, t, m, n))$ for each $c \in C$ and $t \in \mathbb{N}$.
7. $\forall c, t \in C, m \in \mathbb{N} \ (\text{power}(c, t, m, 0) \rightarrow \text{power}(c, t, m, 0))$ for each $c \in C$ and $t \in \mathbb{N}$.
8. $\forall c, t \in C, m, n \in \mathbb{N} \ (\text{power}(c, t, m, n) \rightarrow \text{power}(c, t, m, n))$ for each $c \in C$ and $t \in \mathbb{N}$.
9. $\forall c, t \in C, m \in \mathbb{N} \ (\text{power}(c, t, m, 0) \rightarrow \text{power}(c, t, m, 0))$ for each $c \in C$ and $t \in \mathbb{N}$.
10. $\forall c, t \in C, m, n \in \mathbb{N} \ (\text{power}(c, t, m, n) \rightarrow \text{power}(c, t, m, n))$ for each $c \in C$ and $t \in \mathbb{N}$.
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 [KT09], 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 \( \text{exp}(e(x, y), 0, \ldots, 0) \). Metatheoretically, the terms \( \text{exp}(e(x, y), 0, \ldots, 0) \) and \( e(x, y) \) denote the same quantity, but \( \text{exp}(e(x, y), 0, \ldots, 0) \neq e(x, y) \). It is not a problem for the intruder rules since he has rules for transforming \( \text{exp}(e(x, y), 0, \ldots, 0) \) to \( e(x, y) \), but some congruences within the initial protocol \( T \) would be lost. For example, there would be no syntactical equivalence between \( e(x, y) \uparrow c_1 \uparrow c_1 = e(x, y) \) and \( e(x \star c_1, y \star c_1^{-1}) = e(x \star y, 0, \ldots, 0) \). 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 [KT09]. ProVerif is able to handle the encoded rules without difficulty.

Any clause \( r_1, \ldots, r_n \rightarrow r_0 \) from a theory \( T \) is encoded by substituting all non-ground, non-standard subterms with their \( C \)-exponent-ground encodings. Similarly to [KT09], the encoded clause is

\[
\text{⌜ } \theta(r_1) \text{⌝}, \ldots, \text{⌜ } \theta(r_n) \text{⌝}, C \rightarrow \text{⌜ } \theta(r_0) \text{⌝},
\]

where \( \theta \) is the substitution from non-ground non-standard subterms to new variables, and \( C \) 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:

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

The set of clauses \( C \) is defined as follows:

\[
C = \{ M(\text{⌜ } \theta(s) \text{⌝}, c, x_{s \star c}) | s \star c \in \mathcal{R} \} \\
\cup \{ E(\text{⌜ } \theta(s) \text{⌝}, c, x_{s \uparrow c}) | s \uparrow c \in \mathcal{R} \} \\
\cup \{ P(\text{⌜ } \theta(s_1) \text{⌝}, \text{⌜ } \theta(s_2) \text{⌝}, x_{e(s_1, s_2)}) | e(s_1, s_2) \in \mathcal{R} \}.
\]

We see that the substitution \( \theta \) and encoding \( \text{⌜ } \cdot \text{⌝} \) 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 \( E(\langle t^\uparrow, c, \uparrow t \uparrow c^\downarrow \rangle) \), \( M(\langle t^\uparrow, c, \uparrow t \uparrow c^\downarrow \rangle) \), \( P(\langle t^\uparrow, c, \uparrow t \uparrow c^\downarrow \rangle) \) 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, \( E(t, s, s \ast s) \) 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 [KT09].

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

- \( E(\langle t^\uparrow, c, \uparrow t \uparrow c^\downarrow \rangle) = E(\langle t^\uparrow, c, \text{incr} \exp_i(\langle t^\uparrow \rangle) \rangle) \). This is an instance of the rule (17) for some \( i \) where \( c = c_i \).
- \( E(\langle t^\uparrow, c, \uparrow t \uparrow c^\downarrow \rangle) = E(\langle t^\uparrow, c, \text{decr} \exp_i(\langle t^\uparrow \rangle) \rangle) \). This is an instance of (18) for some \( i \) where \( c = c_i \).
- \( M(\langle t^\uparrow, c, \uparrow t \uparrow c^\downarrow \rangle) = M(\langle t^\uparrow, c, \text{incr} \mult_i(\langle t^\uparrow \rangle) \rangle) \). This is an instance of (19) for some \( i \) where \( c = c_i \).
- \( M(\langle t^\uparrow, c, \uparrow t \uparrow c^\downarrow \rangle) = M(\langle t^\uparrow, c, \text{decr} \mult_i(\langle t^\uparrow \rangle) \rangle) \). This is an instance of (20) for some \( i \) where \( c = c_i \).
- If the head symbols of both \( s \) and \( t \) are not \( \ast \), then 
  \( P(\langle t^\uparrow, s^\uparrow, \exp(e(t, s)) \rangle) = P(\langle t^\uparrow, s^\uparrow, \text{incr} \exp_i(\langle t^\uparrow \rangle) \rangle) \), which is an instance of (21).

Let \( C \) 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 \( t \ast C \) denote the term \( t \ast c_{i1} \ast \cdots \ast c_{ik} \) (it is well-defined up to \( \sim \)) and \( t \uparrow C \) denote the term \( t \uparrow c_{i1} \uparrow \cdots \uparrow c_{ik} \). We also define the metatheoretical functions \( \text{incr}^X_C \), where \( X \in \{\text{mult}, \text{exp}\} \), as follows:

\[
\begin{align*}
\text{incr}^X_{\emptyset}(t) &= t \\
\text{incr}^X_{C \cup \{c\}\emptyset}(t) &= \text{incr}^X_{c}(\text{incr}^X_{C}(t)) & c \in C \\
\text{incr}^X_{C \cup \{c\}\emptyset^{-1}}(t) &= \text{decr}^X_{c}(\text{incr}^X_{C}(t)) & c \in C .
\end{align*}
\]

Again, the functions \( \text{incr}^\mult_C \) and \( \text{incr}^\exp_C \) are well-defined due to the commutation properties of the functions \( \text{incr}^X \) and \( \text{decr}^X \). We use the defined notions to treat more complex cases of applying the predicate \( P \).

- If the head symbol of \( s \) is not \( \ast \), then 
  \( P(\langle t^\uparrow, s^\uparrow, \exp(e(t, s)) \rangle) = P(\langle t^\uparrow, \text{incr}^\mult_C(\langle s^\uparrow \rangle), \text{incr}^\exp_C(\langle e(t^\uparrow, s^\uparrow) \rangle) \rangle) \), which is an instance of (22), according to the definition of \( \text{incr}^\exp \) and \( \text{incr}^\mult \).
- If the head symbol of \( t \) is not \( \ast \), then 
  \( P(\langle t^\uparrow, s^\uparrow \ast C, \exp(e(t, s)) \rangle) = P(\langle t^\uparrow, \text{incr}^\mult_C(\langle s^\uparrow \rangle), \text{incr}^\exp_C(\langle e(t^\uparrow, s^\uparrow) \rangle) \rangle) \), which is an instance of (23).
- \( P(\langle t^\uparrow, C_1, \exp(e(t, s)) \rangle) = P(\langle \text{incr}^\mult_C(\langle t^\uparrow \rangle), \text{incr}^\mult_C(\langle s^\uparrow \rangle), \text{incr}^\exp_C(\langle e(t^\uparrow, s^\uparrow) \rangle) \rangle) \). Therefore it can be derived from (24).
exponentiation nor the bilinear pairing operations, then \( A \) is a non-ground term.

Example 6. For this clause, we get the rule
\[ R = I(e(x, y \ast a)) \rightarrow I(secret). \]

Let \( C = \{c_1, c_2\} \). Let \( secret \) be a constant of \( T \). Suppose that \( T \) contains the fact
\[ R = I(e(x, y \ast a)) \rightarrow I(secret). \]

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, \( T \cup T_E \vdash_e \neg b \).

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 \( T \cup T_E \vdash_e b \) obtained using C-exponent-ground substitutions, then \( T_C \vdash e b \).
Proof:
The proof of the lemma contains a larger number of different cases than the similar proof in [KT09]. It shows that using facts and rules encoded by \( ^r \cdot \gamma \) one can derive all the \( \gamma \)-exponent-ground atoms that can be derived by \( T \cup T^E \), and the only difference is that the derived term will be encoded.

Let \( \pi = b_1, \ldots, b_l \) be a \( \gamma \)-exponent-ground derivation for \( T \cup T^E \), \( \vdash \sim \) \( b \) obtained using \( \gamma \)-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 \( T \cup T^E \) using a \( \gamma \)-exponent-ground substitution \( \sigma \). It is enough to show that \( ^r \gamma b \) can be syntactically derived from \( \pi_{<l} \) using \( T_C \). There are two cases to consider:

1. If \( b \) is obtained using a clause of \( T^E \), then \( b = I(t) \) for some \( \gamma \)-exponent-ground term \( t \). There are three subcases:
   - (a) The set \( \pi_{<l} \) contains atoms \( I(r) \) for a \( \gamma \)-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 \( I(\gamma t) \) can be obtained from \( I(\gamma r) \) and \( I(\gamma c_i) \) using the following clauses:
     - (5) if the reduced form of \( r \) is standard.
     - (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 \( \gamma \)-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 \( I(\gamma t) \) can be obtained from \( I(\gamma r) \) and \( I(\gamma c_i) \) using analogical clauses that are defined for multiplication in \( T_C \).
   - (c) The set \( \pi_{<l} \) contains atoms \( I(r) \) and \( I(s) \) for \( \gamma \)-exponent-ground \( r \) and \( s \), such that \( t \sim c(r,s) \). The atom \( I(\gamma t) \) can be obtained from \( I(\gamma r) \) and \( I(\gamma s) \) using the following clauses:
     - (i) If \( r \) (or \( s \)) is not of the form \( \text{mult}(x_0,x_1,\ldots,x_m) \), then (7) must be applied to \( r \) (or \( s \)) in order to get \( I(\text{mult}(r_0,r_1,\ldots,r_m)) \) and \( I(\text{mult}(s_0,s_1,\ldots,s_m)) \), where \( r \sim \text{mult}(r_0,r_1,\ldots,r_m) \) and \( s \sim \text{mult}(s_0,s_1,\ldots,s_m) \).
     - (ii) Apply the rule (16) to \( \text{mult}(r_0,r_1,\ldots,r_m) \) and \( \text{mult}(s_0,s_1,\ldots,s_m) \).

2. If \( b \) is obtained by some \( \gamma \)-exponent-ground clause \( r_1, \ldots, r_n \rightarrow r_0 \) of \( T \), there exists a \( \gamma \)-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 \( ^r \gamma b \) 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):
   
   \[
   \begin{align*}
   & M(\gamma \theta(s_1^r) \uparrow b_1, x_1), \ldots, M(\gamma \theta(s_j^r) \uparrow b_j, x_j), \\
   & E(\gamma \theta(t_1^r) \uparrow d_1, y_1), \ldots, E(\gamma \theta(t_k^r) \uparrow d_k, y_k), \\
   & P(\gamma \theta(u_1^r) \uparrow z_1), \ldots, P(\gamma \theta(u_i^r) \uparrow z_i), \\
   & \gamma \theta(r_1), \ldots, \gamma \theta(r_n) \rightarrow \gamma \theta(r_0).
   \end{align*}
   \]

Define a substitution \( \sigma^* \), which will be applied to \( R \rightarrow S \) to obtain \( ^r \gamma b \) as follows:

- \( \sigma^*(x) = \gamma \sigma(x) \), for \( x \in \text{var}(r_1, \ldots, r_n) \);
- \( \sigma^*(x_i) = \gamma \sigma(s_i) \);
- \( \sigma^*(y_i) = \gamma \sigma(t_i) \);
\( \sigma^*(z_i) = \tau \sigma(w_i) \).

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 \( \sigma^*(\tau \theta(u)) = \tau \sigma(u) \):

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

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

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

\[
\begin{align*}
E^* & \equiv \sigma^*(E(\tau \theta(t_i), d_i, y_i)), \\
M^* & \equiv \sigma^*(M(\tau \theta(s_i), b_i, x_i)), \\
P^* & \equiv \sigma^*(P(\tau \theta(u_i), \tau \theta(v_i), z_i))
\end{align*}
\]

can all be derived from \( T_C \).

- For exponentiation, we have: \( E^* = \sigma^*(E(\tau \theta(t_i), d_i, y_i)) = E(\sigma^*(\tau \theta(t_i)), d_i, \sigma^*(y_i)) = E(\tau \sigma(t_i), d_i, \tau \sigma(t_i) \uparrow d_i, \tau) \).

  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: \( M^* = M(\tau \sigma(s_i), b_i, \tau \sigma(s_i) \star b_i) \). 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 \( P^* = \sigma^*(P(\tau \theta(u_i), \tau \theta(v_i), z_i)) = P(\tau \sigma(u_i), \tau \sigma(v_i), \tau c(\sigma(u_i), \sigma(v_i))) \).

  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 \( T \cup T_E^\sim \vdash_{\sim} b \). 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.
8 Decoding the Terms: Soundness of the Reduction

We have proved that \( T \cup T_E \vdash \neg b \) implies \( T_C \vdash \neg \psi(b) \), but this is not sufficient. Now we are going to prove that \( T_C \vdash \neg \psi(b) \) implies \( T \cup T_E \vdash \neg b \). 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 \( T \cup T_E \vdash \neg I(u) \). If \( T \) was empty, the verification task would be trivial.

Previously, the function \( t_{2i} \) was defined only on integer terms. Now the domain of \( t_{2i} \) will be extended to all terms in the same way as it has been done in [KT09]. If the term is not an integer term, then \( t_{2i} \) turns it to 0.

- \( t_{2i}(0) = 0 \);
- \( t_{2i}(s(t)) = t_{2i}(t) + 1 \);
- \( t_{2i}(p(t)) = t_{2i}(t) - 1 \);
- \( t_{2i}(t) = 0 \), for any term \( t \notin \{0, s(t'), p(t')\} \) for some term \( t' \).

Now we can define the decoding function, a mapping \( \downarrow \cdot \downarrow \) from terms over \( \Sigma_{pair} \) to terms over \( \Sigma \).

- \( \downarrow x \downarrow = x \), for a variable \( x \);
- \( \downarrow 0 \downarrow = u \);
- \( \downarrow s(t) \downarrow = u \);
- \( \downarrow p(t) \downarrow = u \);
- \( \downarrow exp(t, s_1, \ldots, s_m) \downarrow = \downarrow t \downarrow \uparrow c_1^{t_{2i}(s_1)} \uparrow \cdots \uparrow c_m^{t_{2i}(s_m)} \);
- \( \downarrow mult(t, s_1, \ldots, s_m) \downarrow = \downarrow t \downarrow \star c_1^{t_{2i}(s_1)} \star \cdots \star c_m^{t_{2i}(s_m)} \);
- \( \downarrow f(t_1, \ldots, t_n) \downarrow = f(\downarrow t_1 \downarrow, \ldots, \downarrow t_n \downarrow) \),
  where \( f \notin \{0, s, p, exp, mult\} \);
- \( \downarrow p(t) \downarrow = p(\downarrow t \downarrow) \), 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 [KT09].

Example 7. Let \( C = \{c_1, c_2\} \). Suppose that we are given a term \( t := exp(f(p(0), x), 0, s(s(0))) \) over \( \Sigma_{pair} \), where \( x \) is a variable and \( f \) is arbitrary functional symbol. We have:

\[
\downarrow t \downarrow = \downarrow exp(f(p(0), x), 0, s(s(0))) \downarrow \\
= \downarrow f(p(0), x) \downarrow \uparrow c_2^n \\
= f(\downarrow p(0) \downarrow, \downarrow x \downarrow \uparrow c_2^n \\
= f(\downarrow u \downarrow) \uparrow c_2^n \quad \square
\]
There must be a relationship between the functions $\gamma, \cdot$ and $\cdot, \cdot$. 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 $\gamma t^\cdot$, equivalence modulo $\sim$ is sufficient.

**Lemma 9.** Let $t$ be a $C$-exponent-ground term over $\Sigma$. Then $\gamma t^\cdot \sim t$.

**Proof:**
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 [KT09], but it includes additional cases for pairing.

- If $t' = t_0 \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$ for some integers $k_1, \ldots, k_m$ and a $C$-exponent ground term $t_0$, then $\gamma t' = \exp(\gamma t_0, i2t(k_1), \ldots, i2t(k_m))$. By definition of $\cdot, \cdot$ and the fact that $i2t(i2t(k)) = k$, we obtain $\gamma t^\cdot = \gamma t_0^\cdot \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$. The induction hypothesis yields that $\gamma t_0^\cdot \sim t_0$, and therefore $\gamma t^\cdot \sim t_0 \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$. Hence, $\gamma t^\cdot \sim t'$. Since $t \sim t'$, Lemma 5 implies that $\gamma t^\cdot \sim \gamma t^\cdot$, and so $\gamma t^\cdot \sim \gamma t^\cdot$. Consequently, $t \sim t' \sim \gamma t^\cdot$.\]

- If $t' = t_0 \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$ for some integers $k_1, \ldots, k_m$ and a $C$-exponent ground term $t_0$, then $\gamma t' = \mult(\gamma t_0, i2t(k_1), \ldots, i2t(k_m))$. As in the case of exponentiation, we get $\gamma t^\cdot = \gamma t_0^\cdot \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$. According to induction hypothesis, $\gamma t^\cdot \sim t_0 \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$. Applying Lemma 5, we again get that $t \sim t' \sim \gamma t^\cdot$.

- If $t' = e(r', s')$, then we have three different cases:
  - If $r'$ and $s'$ are standard, then $\gamma t^\cdot = e(\gamma r', \gamma s') = e(\gamma r^\cdot, \gamma s^\cdot) = e(r', s')$. By definition of $\cdot, \cdot$, we get $\gamma t^\cdot = t'$.
  - If either $r' = r_0 \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$ or $s' = s_0 \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$, then $\gamma t^\cdot = \exp(\gamma r^\cdot, \gamma s^\cdot, i2t(k_1), \ldots, i2t(k_m))$ or $\gamma t^\cdot = \exp(\gamma r^\cdot, \gamma s_0, i2t(k_1), \ldots, i2t(k_m))$.
  - By definition of $\cdot, \cdot$, we obtain:
    - $\gamma t^\cdot = e(\gamma r^\cdot, \gamma s^\cdot) \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$ or
    - $\gamma t^\cdot = e(\gamma r_0^\cdot, \gamma s^\cdot) \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$.

The induction hypothesis yields that $\gamma r_0^\cdot \sim r_0$, $\gamma s_0^\cdot \sim s_0$, $\gamma r^\cdot \sim r'$, and $\gamma s^\cdot \sim s'$. We get:

- $\gamma t^\cdot = e(r', s_0) \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$ or
- $\gamma t^\cdot = e(r_0, s') \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$.

We get that $\gamma t^\cdot \sim t'$, and, according to Lemma 5, $t \sim \gamma t^\cdot$.

- If both $r' = r_0 \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$ and $s' = s_0 \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$, then $\gamma t^\cdot = \exp(\gamma r_0, \gamma s_0, i2t(k_1 + l_1), \ldots, i2t(k_m + l_m))$.

Again, by definition of $\cdot, \cdot$, we get that:

- $\gamma t^\cdot = e(\gamma r_0, \gamma s_0, c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_1) + l_1})$. 23
Lemma 10. Let without the proof. Indeed represents exponentiation, multiplication, or pairing). There is an analogous lemma in [KT09] without the proof.

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 [KT09] 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 $\llangle s \rrangle \sim \llangle t \rrangle \uparrow d$.
- If $M(t,d,s)$ can be derived from $T_C$, then $d \in C \cup C^{-1}$ and $\llangle s \rrangle \sim \llangle t \rrangle \times d$.
- If $P(r,s,t)$ can be derived from $T_C$, then $\llangle t \rrangle \sim e(\llangle r \rrangle, \llangle s \rrangle)$.

Proof: The proof of each point can be carried out by case distinction.

Exponentiation: The variable $d$ belongs to the set $C \cup C^{-1}$ because the rules for predicate $E$ are all only of the form $E(t,c_i, incr_i^{exp}(t))$, where $c_i \in C$, or of the form $E(t,c_i, decr_i^{exp}(t))$, where $c_i \in C^{-1}$. The proof of equivalence can be carried out by case distinction.

- $E(t,d,s) = E(t,c_i, incr_i^{exp}(t))$.

Then, $\llangle s \rrangle = incr_i^{exp}(t)$. We have three different cases for $t$:

- Let $t$ be of the form $exp(t_0, \ldots, t_m)$, where $t_i = p(0)$ and $\forall j \neq i t_j = 0$.

We have that $\llangle incr_i^{exp}(t) \rrangle = \llangle t_0 \rrangle$. On the other hand,

$$
\llangle t_0 \rrangle \uparrow d = \llangle t_0 \rrangle \uparrow c_i \\
= exp(t_0, \ldots, t_m) \uparrow c_i \\
= \llangle t_0 \rrangle \uparrow c_1 \uparrow \ldots \uparrow c_m \uparrow c_i \\
= \llangle t_0 \rrangle \uparrow c_1 \uparrow \ldots \uparrow c_i \uparrow \ldots \uparrow c_m \uparrow c_i \\
= \llangle t_0 \rrangle \uparrow c_1 \uparrow \ldots \uparrow c_i \uparrow \ldots \uparrow c_m \uparrow c_i \\
= \llangle t_0 \rrangle .
$$

- Let $t$ be of the form $exp(t_0, \ldots, t_m)$, where either $t_i \neq p(0)$ or $\exists j \neq i t_j \neq 0$. We have that

$$
\llangle incr_i^{exp}(t) \rrangle = exp(t_0, \ldots, t_{i-1}, incr_i(t_i), t_{i+1}, \ldots, t_m) \\
= \llangle t_0 \rrangle \uparrow c_1 \uparrow \ldots \uparrow c_i \uparrow \ldots \uparrow c_m .
$$

On the other hand,
possible uses of the predicate $P$

**Multiplication:** This is analogical to exponentiation. We need to use functions

- **Pairing:**

\[
\left< t, s \right> = \left< \in\text{cr}^E(t), \in\text{cr}^E(s) \right>
\]

- If $t$ be not of the form $\in\text{cr}^E(t_0, \ldots, t_m)$, then, by definition of function $\in\text{cr}^E$, we have

\[
\left< \in\text{cr}^E(t), \in\text{cr}^E(s) \right> = \left< \in\text{cr}^E(\in\text{cr}^E(t), \in\text{cr}^E(s)) \right>
\]

**Multiplication:** This is analogical to exponentiation. We need to use functions $\in\text{cr}^m$ and $\in\text{cr}^m$.

**Pairing:** This equivalence can also be proved by case distinction. We have to look through all possible uses of the predicate $P$.

- $P(r, s, t) = P(x, y, e(x, y))$. This is the simplest case, we have that $\left< t, s \right> = \left< e(x, y), e(x, y) \right>$, directly from the definition of $\left< \cdot, \cdot \right>$.

- $P(r, s, t) = P(\text{mult}(x, x_1, \ldots, x_m), y, e(x, y), x_1, \ldots, x_m)$.

  We have:

  \[
  \left< t, s \right> = \left< \in\text{cr}^E(e(x, y), x_1, \ldots, x_m) \right>
  \]

  \[
  = \left< \in\text{cr}^E(e(x, y), x_1) \right> \uparrow \ldots \uparrow e(12i(x_1)) \uparrow \ldots \uparrow e(12i(x_m))
  \]

  On the other hand,

  \[
  \left< e(x, y), y \right> = \left< e(\text{mult}(x, x_1, \ldots, x_m), y, y) \right>
  \]

  \[
  = \left< e(x, \in\text{cr}^E(x_1), \ldots, \in\text{cr}^E(x_m), y) \right>
  \]

  \[
  \sim \left< e(x, \in\text{cr}^E(x), \in\text{cr}^E(x_1), \ldots, \in\text{cr}^E(x_m), y) \right>
  \]

- $P(r, s, t) = P(x, \text{mult}(y, y_1, \ldots, y_m), e(x, y), y_1, \ldots, y_m)$). The proof is almost the same, and the only difference is that the arguments of $e$ are switched.
\[ P(r, s, t) = P(\text{mult}(x, x_1, \ldots, x_m), \text{mult}(y, y_1, \ldots, y_m), \text{exp}(e(x, y), z_1, \ldots, z_m)) \]

which requires the predicates \(A(x_1, y_1, z_1), \ldots, A(x_m, y_m, z_m)\) to be true.

According to the definition of the predicate \(A(\ldots)\), we have \(\forall i \; z_i = i2t_i(x_i) + t2i(y_i)\).

On the other hand,

\[
e^{(x, y)} = e^{(\text{mult}(x, x_1, \ldots, x_m))} \cup e^{(\text{mult}(y, y_1, \ldots, y_m))} \cup \ldots \cup e^{(t2i(x_1) + t2i(y_1))} \cup \ldots \cup e^{(t2i(x_m) + t2i(y_m))}.
\]

\[
e^{(x, y)} = e^{(\text{mult}(x, x_1, \ldots, x_m))} \cup e^{(\text{mult}(y, y_1, \ldots, y_m))} \cup \ldots \cup e^{(t2i(x_1) + t2i(y_1))} \cup \ldots \cup e^{(t2i(x_m) + t2i(y_m))}.
\]

\[
P(r, s, t) = P(\text{mult}(x, x_1, \ldots, x_m), \text{mult}(y, y_1, \ldots, y_m), e(x, y)).
\]

Example 8. Let \(C = \{e_1, e_2\}\). According to the definition of \(T_C\), the fact

\[
P(x, \text{mult}(y, y_1, y_2), \text{exp}(e(x, y), y_1, y_2)),
\]

where \(x, y, y_1, y_2\) are variables, is a fact of \(T_C\).

Consider substitution

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

for a constant \(g\). Then,

\[
P(g, \text{mult}(g, s(0), g), \text{exp}(e(g, g), s(0), g))
\]

is an instance of the previous fact. We have that

\[
\text{exp}(g, g), s(0), g) \cup e(g, g) \cup c_1
\]

26
Lemma 11. Let \( a = p(t) \) be an atom, such that \( p \) occurs in \( T \). Then, \( T_C \vdash_C a \) implies \( T \cup T_E \vdash_\sim a \).

**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 \( \vdash a_i \). The proof can be carried out by induction and case distinction, in the same way as it has been done in [KT09]. There are additional cases for pairing.

Let \( \pi = a_1, \ldots, a_t \) 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 \( \vdash a_{l,j} \) can be derived from \( \vdash a_{<l,j} \), where \( a_{<l} = a_1, \ldots, a_{l-1} \), and \( a_{<l,j} \) is the sequence of atoms obtained from \( a_{<l} \) by removing all atoms of the form \( E(\ldots), M(\ldots), \) and \( P(\ldots) \), by replacing all the remaining atoms \( a_i \) by \( \vdash a_i \).

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

Here we list all the possible cases that we have to consider, including those that have been proved in [KT09].

1. If \( a_i \) is obtained using the integer derivation rules (2)—(4), it must be of the form \( I(0) \), \( I(s(t)) \), or \( I(p(t)) \). Therefore, \( \vdash a_{l,j} = u \), and we have \( T \cup T_E \vdash_\sim I(u) \).

2. If \( a_i \) is obtained using the rules (4)—(8), it is enough to note that \( \vdash t_i = \vdash \text{exp}(t, 0, \ldots, 0)_{\vdash} = \vdash \text{mul}(t, 0, \ldots, 0)_{\vdash} \).

3. If \( a_i \) is obtained using the rule (9), the atom \( a_i \) is of the form

\[
I(\exp(s_0, \ldots, s_{l-1}, s'_l, s_{l+1}, \ldots, s_m)),
\]

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, \( \vdash b_{\vdash} = I(\vdash \text{exp}(s_0, \ldots, s_m), \vdash) \) and \( \vdash I(c_i)_{\vdash} = I(c_i) \) are elements of \( \vdash \pi_{<l,i} \).

Now we need to derive \( \vdash a_{l,j} \) from \( \vdash b_{\vdash} \) and \( I(c_i) \).

- If \( 2i(s'_i) > 2i(s_i) \), apply the clause \( I(x), I(y) \rightarrow I(x \uparrow y) \) from \( T_E \) \( t2i(s'_i) - t2i(s_i) \) times.
- If \( 2i(s'_i) < t2i(s_i) \), apply the clause \( I(x) \rightarrow I(x^{-1}) \) to \( I(c_i) \), then apply the rule \( I(x), I(y) \rightarrow I(x \uparrow y) \) from \( T_E \) \( t2i(s_i) - t2i(s'_i) \) times.
- If \( 2i(s'_i) = t2i(s_i) \), then \( \vdash a_{l,j} \) is a repetition of \( \vdash b_{\vdash} \).

4. If \( a_i \) is obtained using the rule (10) the proof is analogous to exponentiation.

5. If \( a_i \) is obtained using the rule (16) then the atom \( a_i \) is of the form

\[
\begin{align*}
e((g \downarrow g) \downarrow \text{mult}(g, s(0), g)) &= e(g \downarrow g) \downarrow c_1 \downarrow e_2(s(0)) \downarrow e_2(g) \\
&\quad = e(g, g \downarrow c_1 \downarrow e_2(g) \\
&\quad = e(g, g \downarrow c_1) \\
&\quad \sim e(g, g) \uparrow c_1.
\end{align*}
\]
Suppose that \( \exp(e(u_0, v_0), s_1, \ldots, s_m) \), such that \( I(\text{mult}(u_0, u_1, \ldots, u_m)) \) and \( I(\text{mult}(v_0, v_1, \ldots, v_m)) \) occur in \( \pi_{<I} \), 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 \( s_i = i2((2i(u_1) + 2i(v_1))) \) for each \( i \).

Now denote \( b_1 = I(\text{mult}(u_0, u_1, \ldots, u_m)), b_2 = I(\text{mult}(v_0, v_1, \ldots, v_m)) \). We get that 
\[
\cup_{b_1, \ldots} = I(\cup(\text{mult}(u_0, u_1, \ldots, u_m)), \ldots) \text{ and } b_2 = I(\cup(\text{mult}(v_0, v_1, \ldots, v_m)), \ldots) \text{ are elements of } \cup_{\pi_{<I}}.
\]
Then, \( \cup_{a_{t, j}} \) can be derived from \( \cup_{b_1, \ldots} \) and \( \cup_{a_{t, j}} \) by applying the rule \( I(x), I(y) \rightarrow I(e(x, y)) \) from \( T_E \).

By the above, we have that \( \sigma^{\ast}(r_i) \sim \cup_{a_{t, j}}(\theta(r_i)) \) \((r_i \text{ can not be of the form } w^{-1} \text{ since it is } C\text{-exponent-ground})\). We have that all the \( \cup_{a_{t, j}}(\theta(r_i)) \) for all \( i \in \{1, \ldots, n\} \) are in \( \cup_{\pi_{<I}} \), which means that we can apply the clause \( r_1, \ldots, r_n \rightarrow r_0 \) with \( \sigma^{\ast} \) to obtain \( \sigma^{\ast}(r_0) \sim \cup_{a_{t, j}}(\theta(r_0)) \).
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 $\tilde{a} \equiv I(exp(e(g, g), 0, s(0)))$ has been derived from $T_C$ using the rule

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

Assume that $T_C$ contains a fact

$$I(f(mult(g, p(0), 0), mult(g, 0, s(0)))),$$

and the substitution obtained in the derivation process is

$$\sigma = \{mult(g, p(0), 0)/x, mult(g, 0, s(0))/y, mult(g, s(0), s(0))/z, \exp(e(g, g), 0, s(0))/v\}.$$  

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 $\tilde{a}$ 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. $I(f(mult(g, p(0), 0), mult(g, 0, s(0))))$
2. $I(f(mult(g, p(0), 0), mult(g, 0, s(0))) \rightarrow I(exp(e(g, g), 0, s(0))).$

After decoding the terms, we get:

1. $I(f(g^{*} c_1, g^{*} c_2)) \rightarrow I(e(g, g) \uparrow c_2)$
2. $I(f(g^{*} c_1, g^{*} c_2)).$

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, $I(e(g, g) \uparrow c_2) \sim \tilde{a}$. □

**Proof of Theorem 7:** We have used Lemma 8 to prove that $T \cup T_E \vdash \sim b$ implies $T_C \vdash \sim c \vdash b$. Now suppose that $T_C \vdash \sim c \vdash b \vdash$. By assumption, $b = p(t)$, where $p$ occurs in $T$. Lemma 11 implies that $T \cup T_E \vdash \sim \vdash c \vdash b \vdash$. By Lemma 9, $\vdash c \vdash b \vdash \sim b$, 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 $T \cup T'_E$ without any loss of information.

9 Experiments

We have extended the Horn theory transformer by Küsters and Truderung [KT09] 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 [Jou00] 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 [CVC04] of Shim’s three-party certificate-based one-pass key-exchange protocol [Shi03]. 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
\[
  t_1 = e(t_{11}, t_{12}) \quad \text{and} \quad t_2 = e(t_{21}, t_{22}),
\]
are equal. In symbolic model, obviously the equality \( \sim \), 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 [ARP03] includes the public keys in the generated session key. It was found to be secure.

• **TAK 2.** This protocol by the same authors [ARP03], although similar to the previous one, is insecure [SW07]. 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 [ARP03]. It was found to be secure.


**Efficiency**

We have tested the running time of the analysis on the listed protocols. Similarly to [KT09], 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 \( \left\langle \cdot, \cdot \right\rangle \) 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.

**10 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 [KT09], 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 exponentia-
ation 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 [CC10].

We will now extend the signature Σ with addition ‘+’ in G_1 and multiplication ‘·’ in G_T, intro-
ducing the equations and intruder theory which make them behave as the corresponding operations
in cyclic groups. If ≈ is the new equivalence of terms, and T_{E3} 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 ∪ T_{E3} ⊢_≈ a implies
T_P ∪ T_E ⊢_≈ a. Hence these operations do not help the adversary.

The equations involving + and · 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.

\[
\begin{align*}
x + y & \sim^\sharp y + x \\
(x + y) + z & \sim^\sharp x + (y + z) \\
(x + y)^{-1} & \sim^\sharp x^{-1} + y^{-1} \\
x + 0 & \sim^\sharp x \\
(x + y) & \sim^\sharp (x \star z) + (y \star z) \\
(x \cdot y) & \sim^\sharp (x \uparrow z) \cdot (y \uparrow z)
\end{align*}
\]

(27)

The additions to the intruder theory due to these operations are straightforward. The theory
T_{E3} consists of the rules of the theory T_E, plus the following rules:

\[
I(x), I(y) \rightarrow I(x + y) \quad I(x), I(y) \rightarrow I(x \cdot y)
\]

As the protocol P does not contain + or ·, 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 ·, then any
derivation where + and · 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 · operations. We
call a term over Σ 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 ∗ and ↑, then
the inversions, and finally the multiplication-like operations + and ·. Each term can be brought
to a reduced form and the reduced form is determined uniquely modulo the associativity and/or
commutativity of ∗, ↑, +, · 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:

31
\[ t = e((p \ast a + p \ast b) \ast c, p \ast a \ast c) \uparrow a^{-1} \]
\[ \sim^3 e(p \ast a \ast c + p \ast b \ast c, p \ast a \ast c) \uparrow a^{-1} \]
\[ \sim^3 e(p \ast a \ast c + p \ast b \ast c, p) \uparrow a^{-1} \uparrow a \uparrow c \]
\[ \sim^3 e(p \ast a \ast c + p \ast b \ast c, p) \uparrow c \]
\[ \sim^3 (e(p \ast a \ast c, p) \cdot e(p \ast b \ast c, p)) \uparrow c \]
\[ \sim^3 e(p \ast a \ast c, p) \uparrow c \cdot e(p \ast b \ast c, p) \uparrow c \]
\[ \sim^3 e(p, p) \uparrow c \uparrow a \uparrow c \cdot e(p, p) \uparrow b \uparrow c \]
\[ \sim^3 e(p, p) \uparrow a \uparrow c^2 \cdot e(p, p) \uparrow b \uparrow c^2 \]

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

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

We use the functions \( g^+ \) and \( \hat{g} \) to select addends from sums and factors from products.

For a term \( t \) in reduced form and a pair of functions \((g^+, \hat{g}) \in \text{Sel}^\rightarrow \), we define a term \( DF(t, g^+, \hat{g}) \) as follows:

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

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

Depending on the definition of $g^+$, we select either the first or the second argument of addition.

If we select $p \star a$:

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

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

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

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 $T_E$ 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 $R ≜ r_1, \ldots, r_k → r$ be a rule in the theory $T_P \cup T_E$. Let $a_1, \ldots, a_k → a$ be a ground instance of this rule. Let $(g^+, \hat{g}) ∈ \text{Sel}^T$. Then $DF(a_1, g^+, \hat{g}), \ldots, DF(a_k, g^+, \hat{g}) → DF(a, g^+, \hat{g})$ is an instance of $R$.

Proof:
Let $x_1, \ldots, x_j$ be the variables occurring in the rule $R$. Let $θ$ be the substitution of $x_1, \ldots, x_j$ with ground terms, such that $r, θ ∼^T a_i$ and $rθ ↘ a_i$. Define the substitution $\bar{θ}$ by $x_i\bar{θ} = DF(x_iθ, g^+, \hat{g})$. One can easily verify that $r, θ = DF(a_1, g^+, \hat{g})$ and $r\bar{θ} = DF(a, g^+, \hat{g})$.

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 $(g^+, \hat{g}) ∈ \text{Sel}^T$:

$$T_P ∪ T_{E^1} ⊢ _{\sim} a \implies T_P ∪ T_E ⊢ _{\sim} DF(a, g^+, \hat{g}) .$$

Proof:
The lemma is proved by induction over the derivation length.

We need to show how to obtain the derivation modulo $\sim$ for each $DF(a, g^+, \hat{g})$.

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 $π = b_1, \ldots, b_l$ be a derivation for $T_P ∪ T_{E^1} ⊢ _{\sim} a$ where $a ≈^T b_1$. The proof is based on induction over the length of $π$.

- **Base:** If $l = 0$, there is no derivation. Since $T_P$ does not contain operations $+$ and $\cdot$ and the rules of $T_{E^1}$ have not been used yet, it means that $a$ contains neither $+$ nor $\cdot$. By definition of $DF$, we have $DF(a, g^+, \hat{g}) = a$.

- **Step:** Let $π_{<l} = b_1, \ldots, b_{l-1}$. By the assumption of the lemma, we know that $a ≈^T b_1$ can be derived from $π_{<l}$ by applying a clause from $T_P ∪ T_{E^1}$. We need to show that for any $(g^+, \hat{g}) ∈ \text{Sel}^T$, the term $DF(a, g^+, \hat{g})$ can be derived from $π_{<l}$ modulo $\sim$ using $T_P ∪ T_E$.

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

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

   $$I(x), I(y) → I(x + y) .$$

   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 $π_{<l}$. It is possible that the head operation of $r$ or $s$ is $\cdot$. Let $r = r_1 + \ldots + r_k$ and $s = s_1 + \ldots + s_k$, where $k_r, k_s ≥ 1$ and the head operations of $r_1, \ldots, r_{k_r}, s_1, \ldots, s_{k_s}$ are not $\cdot$.

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

   By induction hypothesis, the atom $I(DF(r_i, g^+, \hat{g}))$ can be derived from $T_P ∪ T_E$ modulo $\sim$, concluding the induction step.

34
(b) Suppose that \( a \) is obtained using the rule
\[
I(x), I(y) \rightarrow I(x \cdot y).
\]
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 \( R \equiv r_1, \ldots, r_k \rightarrow r \) from \( T_P \cup T_E \).

In the derivation of \( a \), we use the instance of this rule \( a_1, \ldots, a_k \rightarrow a \), where \( a_i \) are atoms derived from \( \pi_{x,j} \). 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 \( T_P \cup T_E \).

11 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. [Sma02]) make use of it. While the full treatment of addition is most likely intractable, we may try to adapt some ideas of Kremer and Mazare [KM10] 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 [Möd11] 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 \( \ast \) 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 [BCK09, BMS06]. The results of Kremer and Mazare [KM10] 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 [CW05] should be possible.

References


