View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by DSpace at Tartu University Library

University of Tartu
Faculty of Mathematics and Computer Science
Institute of Computer Science

Alisa Pankova

Bilinear Mappings in Formal Cryptography

Bachelor's thesis (6ECTS)

Supervisor: Peeter Laud

Author: .... "....." June 2011
Supervisor: .... "....." June 2011

Allowed to defend
Professor: .... "....." June 2011

Tartu 2011

Contents

Abstract

Bilinear mappings are quite powerful mathematical structures that can be used in cryptography. They allow constructing cryptographic primitives that would be otherwise ineffective or even impossible. In formal cryptography, the protocols are based on term algebras and process calculi, and can be represented through Horn clauses for analysis purposes. The security of these protocols can be tested with analyzers based on resolution methods. However, there are problems with realization of arithmetic operations. It is easy to compute g^a if the values of both g and a are known, but the values are usually undefined in the protocols. Some research works have been written about the representation of exponentiation in formal model, but there are still many things that should be done. In this work, an attempt to implement an analysis of bilinear mappings in formal cryptography has been done.

1 Introduction

The security properties of the cryptographic primitives are mostly based on mathematical problems that presumably cannot be solved in polynomial time. One of such problems is the discrete logarithm problem. Let $G = \langle P \rangle$ (G is generated by P) be a multiplicative group of order n. Hardness of the discrete logarithm in group G means that given the generator P, size of the group n, and some group element P^a, it is impossible to find the exponent a in polynomial time. A related problem is the Diffie-Hellman problem: given P, P^a and P^b, find P^{ab}. This problem may be used for key exchange between two parties.

Let A and B be the two parties. Let $G = \langle P \rangle$ be the group of order n. First, each party generates a random number from $\mathbb{Z}_n$. A generates an integer a; B generates an integer b. Then they exchange the following values:

- $A \to B$: P^a
- $B \to A$: P^b

Each party may now derive a key P^{ab} that can be used as a new secret key for communication between A and B. The party A calculates $(P^b)^a = P^{ba} = P^{ab}$, and the party B similarly $(P^a)^b = P^{ab}$. If the intruder eavesdrops on the network and gets P^a and P^b, he cannot derive the secret key P^{ab} if the Diffie-Hellman problem cannot be solved in the group G.

The DH problem can be extended to groups with bilinear mappings [7]. This allows the construction of very efficient cryptographic primitives.

Let n be a prime number. Consider two groups of the order n: G_1 and G_T. Let G_1 be an additive group generated by P. Let G_T be a multiplicative group. A mapping e can be defined:

$$e : G_1 \times G_1 \to G_T \ ,$$

which satisfies the following properties:

1. Bilinearity: $\forall R, S, T \in G_1, e(R + S, T) = e(R, T)e(S, T)$, and $e(R, S + T) = e(R, S)e(R, T)$.
2. Non-degeneracy: $e(P, P) \neq 1$.
3. Computability: e is easy to compute.

According to bilinearity, the generator of G_T is the value $e(P, P)$. This fact makes the property of non-degeneracy very important. If the generator of the group G_T was 1, then the whole group would consist only of a single element 1, and it would make no use in cryptography. The computability is necessary since otherwise the mapping would have no practical use.

Another way to define bilinearity is the following equation:

$$\forall R, S \in G_1, \forall a, b \in \mathbb{Z}, e(aR, bS) = e(R, S)^{ab} \ .$$

This definition is better to use if we allow only products in exponents.

For cryptography, the most interesting case is when the discrete logarithm problem is hard in the group G_1. According to bilinearity, the discrete logarithm problem should also be hard in the group G_T. If it did not, then we could use G_T for finding the discrete logarithms of G_1 elements. Given $aP \in G_1$, we might apply the bilinear function e to aP and P. The result would be $e(aP, P) = e(P, P)^a$, and if the discrete logarithm problem could be solved in G_T, we would get the a and thus solve the discrete logarithm problem in G_1.

A new problem can be defined (Bilinear Diffie-Hellman Problem): given aP, bP, cP, and P, find $e(P, P)^{abc}$. This problem cannot be solved if the discrete logarithm problem is hard in G_1.

For cryptography, it is very important that such groups and appropriate bilinear mappings do exist in reality (like Weil and Tate pairings over elliptic curves). These things are of practical use and can be applied in real life.

2 Formal View of Cryptography

There are two main views of cryptography:

1. Formal (Dolev-Yao) view.
   - Messages are elements of term algebra.
   - Possible operations on terms are enumerated.
   - Protocol is represented through a process calculus or a theory.
2. Computational view.
   - Messages are bit strings.
   - Possible operations on bit strings - everything in Probabilistic Polynomial Time.
   - Protocol is a set of probabilistic interactive Turing machines.

The computational view is close to the real world, but it is much more difficult to analyze the security properties of a protocol.

The formal view is simpler to analyze. We may rather easily describe the protocol and the set of possible operations that may be applied on the terms. In Dolev-Yao model, everything can be represented as terms:

- **Hashing**: $h(S)$ is the term that represents the result of applying a hash function h to some bit string S. We do not have to think on what this hash function actually does. It is sufficient to state that if the intruder gets the term $h(S)$, he cannot extract S from it (perfect cryptography assumption), and we have it by default if we do not write additional rules for it. The property that if $S_1 \neq S_2$, then $h(S_1) \neq h(S_2)$ (collision-resistance) is assumed by default, since these two terms are syntactically different (if we do not factorize the terms in such a way that $h(S_1)$ and $h(S_2)$ are equivalent).
- **Encryption**: $enc(K, M)$ is the term that represents the message M encrypted by the key K. We may define different functions for different kinds of encryption. Additionally, we need to define the rules for decryption with an appropriate key:
  - $dec(K, enc(K, M)) = M$ for the symmetric encryption (decrypt with the same key).
  - $dec(K, enc(pk(K), M)) = M$ for asymmetric encryption. Here we use a term $pk(K)$ which represents a public key that corresponds to the secret key K.

- **Signing:** $sign(K, M)$ is the message M signed by the key K. As in the case of encryption, we need rules for ejecting the message out of the signature: $readsign(pk(K), sign(K, M)) = M$ allows to read the signed message by using the public key of the signer. There can be different realizations of signing, we may also write something like $verify(pk(K), sign(K, M)) = true$.

These examples are just standard cryptographic operations, and we may define more functions that will be used in the particular protocol.

There are different automatic protocol analyzers that can be used for verifying the secrecy of protocols in the formal model. One of them is ProVerif [2]. It supports description of a protocol in two formats: Horn clauses and pi-calculus. A protocol defined in pi-calculus is internally also transformed to Horn clauses.

A Horn clause is a logical implication where $F_1, \ldots, F_n$ and G are atoms:

$$F_1 \,\&\, \ldots \,\&\, F_n \to G \ \ ,$$

which means that if the claims $F_1, \ldots, F_n$ are true, then G is also true.

In cryptography, a logical clause may represent the fact that the attacker knows some value. We may define a predicate I such that $I(X)$ is true iff the intruder knows the value of X (it is also possible to define several intruders if necessary). The rules that may be applied on terms can be described with Horn clauses. For example:

$$I(key) \,\&\, I(enc(key, mesage)) \to I(message)$$

is the rule that states: if the intruder I has a key and a message encrypted with that key, he may decrypt it and get the message.

There are actually more things that we need to describe: the protocol itself. This can also be done by using Horn clauses. The messages that are moving in the network are sent out by the communicating parties. We may assume that the first-round messages are sent out in any case (otherwise the protocol would not work), and the intruder gets these messages as facts. The messages of the next rounds depend on the messages that the parties received from the previous rounds, and it is the intruder who decides what the particular party receives in the particular session. If a party A, getting some term X, answers with a term Y, we may write it as $I(X) \to I(Y)$, since if the intruder has X, he may send it to A and get the Y. The exact contents of X and Y (evaluations of their free variables) are determined by the substitution that is applied to both of them at the same time.

Sometimes it is difficult to write a protocol as a set of Horn clauses, especially if it uses many rounds. A protocol can be described as some process calculus, where we may define separate processes for different parties. ProVerif allows describing a protocol in pi-calculus, but afterwards it translates the processes into Horn clauses, so that pi-calculus is just an auxiliary method for describing a protocol.

It is important that two syntactically different terms may actually be cryptographically equivalent. For example, in the case of bilinear mappings, we know that $e(x, y) = e(y, x)$. If we want correct analysis of our protocol, we need to use equational theories. There may exist vulnerabilities that are caused by equivalence of some terms.

An equational theory has been developed for Diffie-Hellman exponentiation in formal model [6]. It allows only ground exponents in terms and only multiplication of exponents, but for some protocols it is sufficient. This work is based on the following equations:

- $(x \uparrow y) \uparrow z = (x \uparrow z) \uparrow y$;
- $(x \uparrow y) \uparrow y^{-1} = x$;
- $(x^{-1})^{-1} = x$.

It is not easy to implement these equations in ProVerif since it does not support associativity in equations. Therefore, a solution was developed that is based on term encoding.
The intruder may perform two operations: exponentiation and inversion.

- $I(x), I(y) \to I(x \uparrow y)$;
- $I(x) \to I(x^{-1})$.

The theory that contains these two rules has been defined as T_{DH}.

Since there is finite number of ground exponents, the intruder may use only predefined grounded exponents that belong to some set C. Since inversion is allowed, the intruder may also exponentiate group elements with inverses of these exponents (the set C^{-1}). If there are only products in exponents, then the rule that allows finding inverses of group elements is unnecessary. This rule would not give the intruder any additional power. Therefore, the rules of intruder will be the following:

- $I(x), I(c) \to I(x \uparrow c)$ for each $c \in C$;
- $I(x), I(c) \to I(x \uparrow c^{-1})$ for each $c \in C$.

This kind theory has been defined as T_{DH}^{C}, and its rules are determined by the set C.

In order to provide syntactical equality, the terms have to be encoded in such a way that congruent terms would have the same encoding. ProVerif should perform the analysis on encoded terms.

The main idea of [6] is to find out all the possible exponents that will be used in the protocol. It was proved that we do not need to consider unbounded number of sessions in case of DH exponentiation, and therefore we may use finite number of exponents.

Let all the possible exponents be $c_1, \ldots, c_m$. Each group element may then be encoded as $exp(g, x_1, \ldots, x_m)$, where $x_1, \ldots, x_m$ are the powers of exponents in the given group element, and g is some group element whose exponents are unknown, or, most probably, is the group generator. For example, having the set of exponents $\{a, b, c\}$, we may encode P^{ab} as $exp(P, 1, 1, 0)$. A more complex term $P^{a^3c^{-1}}$ would then be encoded as $exp(P, 3, 0, -1)$. The problem is that the powers cannot be denoted directly with numbers since ProVerif does not support arithmetic operations. The integers also have to be be encoded: $0 \to 0$, $1 \to s(0)$, $2 \to s(s(0)) \ldots -1 \to p(0)$, $-2 \to s(0) \ldots$. As the result, P^{ab} would then look like $exp(P, s(0), s(0), 0)$, and $P^{a^3c^{-1}}$ like $exp(P, s(s(s(0))), 0, p(0))$.

A set of rules has been defined for exponentiation of the encoded group elements. This kind of encoding allows syntactical equivalence of congruent terms. For example, the encoding of both P^{ab} and $P^{bacc^{-1}}$ will be $exp(P, s(0), s(0), 0)$.

This solution made it possible to test with ProVerif several actual protocols using DH exponentiation, and this work is a good base for construction of a similar theory for bilinear mappings.

3 Pairing-Based Cryptography in Formal Model

3.1 Equational Theory for Bilinear Mappings

If we want to use the properties of bilinear pairings in our protocols, we need to define a set of rules for the intruder. This set of rules is actually a theory. The intruder is permitted to multiply elements of G_1 with integers, apply the pairing function e to elements of G_1, and raise elements of G_T into powers. No addition of exponents is allowed since this theory is based on the [6]'s theory for exponentiation.

Let $\uparrow$ denote the exponentiation and $\star$ the multiplication. The algebraic properties for bilinear mappings can be modelled by congruence relation $\sim$ on terms by following equations:

1. $(x \uparrow y) \uparrow z \sim (x \uparrow z) \uparrow y$ for each $x \in G_T$, $y \in \mathbb{Z}$;
2. $(x \uparrow y) \uparrow y^{-1} \sim x$ for each $x \in G_T$, $y \in \mathbb{Z}$;
3. $(x \star y) \star z \sim (x \star z) \star y$ for each $x \in G_1$, $y \in \mathbb{Z}$;
4. $(x \star y) \star y^{-1} \sim x$ for each $x \in G_1$, $y \in \mathbb{Z}$;
5. $(x^{-1})^{-1} \sim x$ for inverses in G_1, G_T, and $\mathbb{Z}$;.
6. $e(x, y \star z) \sim e(x, y) \uparrow z$ for each $x, y \in G_1$, $z \in \mathbb{Z}$;
7. $e(x, y^{-1}) \sim e(x, y)^{-1}$ for each $x, y \in G_1$;
8. $e(x, y) \sim e(y, x)$ for each $x, y \in G_1$.

The last equation is actually implied by bilinearity property, but we have to add this rule separately. The theory does not contain information about the size of the group or its generator, and ProVerif cannot reduce any expression $e(x, y)$ to the form $e(P, P) \uparrow c$, where P is the generator of G_1 and c is some integer.

In [6], several notions were defined. These definitions will be a little bit different for the theory of bilinear mappings, since we extend the signature and add more rules to the existing theory.

Definition 1 *A term is called reduced, if the '2' cannot be applied from left to right (mod $'1'$), the $'4'$ cannot be applied from left to right (mod $'3'$), the $'7'$ cannot be applied from left to right (mod $'6'$), and $'5'$ cannot be applied from left to right (mod $'1'$), (mod $'3'$), and (mod $'6'$).*

Definition 2 *A term is called standard, if its head symbol is neither $\star$, $\uparrow$, $^{-1}$, nor e.*

Definition 3 *A term is called pure, if the symbols $\star$, $\uparrow$, $^{-1}$, and e do not occur in it.*

After the congruence relations have been described, we need to define rules for the intruder. They are described in the following theory (T_E):

1. $I(x), I(y) \rightarrow I(e(x,y))$ for each $x, y \in G_1$;

2. $I(x), I(y) \rightarrow I(x \star y)$ for each $x \in G_1$, $y \in \mathbb{Z}$;

3. $I(x), I(y) \rightarrow I(x \uparrow y)$ for each $x \in G_T$, $y \in \mathbb{Z}$;

4. $I(x) \rightarrow I(x^{-1})$.

Here the same notation x^{-1} is used for inverses in G_1, G_T, and $\mathbb{Z}$.

Although G_1 is additive and G_T is multiplicative, the intruder has freedom to apply the addition operation to the elements of G_T and multiplication operation to the elements of G_1. This, however, does not give him anything useful, and therefore type constraints can be added to the intruder rules when applying this theory with ProVerif.

Given a protocol P and a message m, the fact that $T_P \cup T_E \nvdash_E I(m)$ means that the intruder cannot get the message m even employing algebraic properties of bilinear mappings. Here $\vdash_E$ denotes derivation *modulo* the congruence $\sim$.

3.2 Exponent-ground Theory for Bilinear Mappings

In [6], the theory for Diffie-Hellman exponentiation T_{DH} is constrained to a theory T_{DH}^C, that can be used only with exponent-ground terms. In the same way, the theory T_E should be constrained to T_E^C. First, we need to define what does it mean for a term to be exponent-ground in the theory T_E.

Definition 4 *A term t is well-formed if every subterm of t of the form s^{-1} only occurs in a context of the form $s' \uparrow s^{-1}$ or $s' \star s^{-1}$ for some s'.*

Definition 5 *A term is 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.*

Definition 6 *A term is C-exponent-ground if it is exponent-ground and has exponents only from the predefined finite set C.*

We need to define a C-exponent ground theory T_E^C that allows multiplication in G_1 and exponentiation in G_T only with ground multipliers (exponents). Let C be the set of ground variables that may be used as multipliers in G_1 and exponents in G_T. The theory contains the following rules:

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

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 G_1 and G_T if we are dealing only with products in exponents, and the rules 3 and 5 actually give 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 ($T \cup T_E \vdash_E a$), then there exists a C-exponent-ground derivation of a.

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

First, we need to define a function δ_C which turns any terms into C-exponent-ground terms. Applying this function to some derivation $T \cup T_E \vdash_E a$ returns a C-exponent-ground derivation $T \cup T_E^C \vdash_E a$.

Let $C^* = C \cup C^{-1}$. The function will be defined by induction:

- $\delta_C(x) = x$ for a variable x;
- $\delta_C(t \uparrow s) = \delta_C(t) \uparrow s$ if $s \in C^*$;
- $\delta_C(t \uparrow s) = \delta_C(t)$ if $s \notin C^*$;
- $\delta_C(t^{-1}) = \delta_C(t)$;
- $\delta_C(t \star s) = \delta_C(t) \star s$ if $s \in C^*$;
- $\delta_C(t \star s) = \delta_C(t)$ if $s \notin C^*$;

- $\delta_C(f(t_1,\dots t_n)) = f(\delta_C(t_1),\dots \delta_C(t_n))$ for $f \notin \{\uparrow, \star, ^{-1}\}$.

If the set C is fixed, we may write simply δ instead of δ_C. In order to prove the Theorem 1, we need to prove several lemmas first.

Lemma 1 *For any set C of pure, ground terms and for every term t we have:*

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

Proof: This lemma summarizes the properties of the function δ_C defined above. The proof of each point is based on induction. We have to look through all the possible cases of application of δ_C.

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

   - $t = x$ for a variable x: $\delta_C(\delta_C(x)) = \delta_C(x) = \delta_C(t)$, according to the definition. This can be considered as the induction basis.
   - $t = t'^{-1}$ for a term t': $\delta_C(\delta_C(t'^{-1})) = \delta_C(t))$.
   - $t = t' \uparrow s$, where t' is some term and $s \in C^*$. $\delta_C(\delta_C(t' \uparrow s)) = \delta_C(\delta_C(t') \uparrow s) = \delta_C(\delta_C(t')) \uparrow s$, and by induction hypothesis it equals $\delta_C(t') \uparrow s = \delta_C(t)$.
   - $t = t' \uparrow s$, where t' is some term and $s \notin C^*$. $\delta_C(\delta_C(t' \uparrow s)) = \delta_C(\delta_C(t')) = \delta_C(t')$ by induction hypothesis. We also have that $\delta_C(t' \uparrow s) = \delta_C(t') = \delta_C(t)$.
   - $t = t' \star s$, where t' is some term and $s \in C^*$. $\delta_C(\delta_C(t' \star s)) = \delta_C(\delta_C(t') \star s) = \delta_C(\delta_C(t')) \star s$, and by induction hypothesis it equals $\delta_C(t') \star s = \delta_C(t)$.
   - $t = t' \star s$, where t' is some term and $s \notin C^*$. $\delta_C(\delta_C(t' \star s)) = \delta_C(\delta_C(t')) = \delta_C(t')$ by induction hypothesis. On the other hand, $\delta_C(t' \star s) = \delta_C(t') = \delta_C(t)$.
   - $t = f(t_1, \dots t_n)$ for some terms $t_1, \dots, t_n$. $\delta_C(\delta_C(f(t_1,\dots t_n))) = \delta_C(f(\delta_C(t_1),\dots \delta_C(t_n))) = f(\delta_C(\delta_C(t_1)),\dots \delta_C(\delta_C(t_n)))$, which according to induction hypothesis equals $f(\delta_C(t_1),\dots \delta_C(t_n)) = \delta_C(f(t_1,\dots t_n)) = \delta_C(t)$.

2. $\delta_C(t)$ 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.
- $t = t' \star s$, where t' is some term and $s \in C^*$. $\delta_C(t' \star s) = \delta_C(t') \star s$. Since t' is C-exponent-ground by induction hypothesis, and $s \in C^*$, the whole term is also C-exponent-ground.
- $t = t' \star s$, where t' is some term and $s \notin C^*$. $\delta_C(t' \star s) = \delta_C(t')$, and $\delta_C(t')$ is C-exponent-ground according to induction hypothesis.
- $t = f(t_1, \dots t_n)$ for some terms $t_1, \dots, t_n$.
  $\delta_C(f(t_1, \dots t_n)) = f(\delta_C(t_1), \dots \delta_C(t_n))$. The arguments of f, namely $\delta_C(t_1), \dots \delta_C(t_n)$, are C-exponent-ground by induction hypothesis, and after applying the function f, the term is still C-exponent-ground.

3. $\delta_C(t) = t$ iff t is C-exponent ground. The function $\delta_C(t)$ transforms all the terms to C-exponent-ground. It implies that, even if t is not C-exponent-ground, the term $\delta_C(t)$ is C-exponent-ground, and therefore cannot be equal to t. It is sufficient 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.
   - $t = t' \uparrow s$, 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.
   - $t = t' \star s$, where t' is some term and $s \in C^*$. $\delta_C(t' \star s) = \delta_C(t') \star s$. 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 $\delta_C(t) = t' \star s$.

- $t = t' \star s$, where t' is some term and $s \notin C^*$. The term t is not C-exponent-ground.
- $t = f(t_1, \dots t_n)$ for some terms $t_1, \dots, t_n$. $\delta_C(f(t_1, \dots t_n)) = f(\delta_C(t_1), \dots \delta_C(t_n))$. The arguments of f, $\delta_C(t_1), \dots \delta_C(t_n)$, should be C-exponent-ground (otherwise t would not be), and, by induction hypothesis, $f(\delta_C(t_1), \dots \delta_C(t_n)) = f(t_1, \dots t_n) = t$. □

Definition 7 *A term t is exponent-reduced, if every subterm s of t, which occurs as an exponent, is reduced.*

Now it is necessary to prove that δ preserves the equivalence on exponent-reduced terms.

Lemma 2 *For all exponent-reduced terms t and s, if $t \sim s$, then $\delta(t) \sim \delta(s)$.*

Proof: It is sufficient to prove this lemma only for the case where s is a reduced form of t. If r is a reduced form of t and $t \sim s$, then $\delta(t) \sim \delta(r)$, and $\delta(r) \sim \delta(s)$. By transitivity of equivalence relation, $\delta(t) \sim \delta(s)$.

Assume that t is exponent-reduced and s is its reduced form. We proceed by induction on the size of t and consider the following cases:

1. If the head symbol of t is neither $\star, ^{-1}$ nor $\uparrow$, the s should have the same head symbol. There are now two cases for t:
   (a) t is a constant or a variable: $\delta(t) = t = s = \delta(s)$.
   (b) t is of the form $f(t_1, \dots, t_n)$: s is also of the form $f(s_1, \dots, s_n)$, where s_i is a reduced form of t_i. By the induction hypothesis, $\delta(t_i) \sim \delta(s_i)$, and therefore $\delta(t) = f(\delta(t_1), \dots, \delta(t_n)) \sim f(\delta(s_1), \dots, \delta(s_n)) = \delta(s)$.
2. If the head symbol of t is $^{-1}$, then $t = u^{-1}$ for some u. There are 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, $\delta(t) = \delta(r) \sim \delta(s)$.
   (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 (otherwise there would be a contradiction with head symbols). By the induction hypothesis, $\delta(u) \sim \delta(w)$, and since $\delta(t) = \delta(u)$ and $\delta(s) = \delta(w)$ (from the definition of δ), we have $\delta(t) \sim \delta(s)$.

3. If the head symbol of t is $\uparrow$, we may 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 $t_i \sim t_j^{-1}$. If we take these two out from t and get t', we have that s is a reduced form of t'. Because t is exponent-reduced, it follows that $t_i = t_j^{-1}$ or $t_i^{-1} = t_j$: we get that $t_i \in C^*$ iff $t_j \in C^*$. We have $\delta(t) \sim \delta(t')$, and by induction hypothesis $\delta(t') \sim \delta(s)$, which implies $\delta(t) \sim \delta(s)$.

   (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, $\delta(t) = \delta(t_0) \uparrow t_{i_1} \uparrow \ldots \uparrow t_{i_k}$, where $t_{i_1}, \ldots, t_{i_k}$ 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 $\delta(s) = \delta(s_0) \uparrow s_{i_1} \uparrow \ldots \uparrow s_{i_k}$. By the induction hypothesis, $\delta(t_0) = \delta(s_0)$. It follows that $\delta(t) \sim \delta(s)$.

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

Lemma 3 *Let C be a set of pure, ground terms. Let t be a C-exponent-ground term, and θ be a substitution. Then $\delta(t\theta) = t\delta(\theta)$.*

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

1. If t is a standard term, $\delta(t\theta) = t\delta(\theta)$. Let it be the induction basis.

2. If $t = s^{-1}$ for some s, then t was 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 $s' \in C^*$. We have $\delta(t\theta) = \delta(s\theta \uparrow s') = \delta(s\theta) \uparrow s'$. By the induction hypothesis, $\delta(s\theta) = s\delta(\theta)$. Thus, $\delta(s\theta) \uparrow s' = s\delta(\theta) \uparrow s' = t\delta(\theta)$.

4. If $t = s \star s'$, it follows that s is C-exponent-ground and $s' \in C^*$. We have $\delta(t\theta) = \delta(s\theta \star s') = \delta(s\theta) \star s'$. By the induction hypothesis, $\delta(s\theta) = s\delta(\theta)$. Thus, $\delta(s\theta) \star s' = s\delta(\theta) \star s' = t\delta(\theta)$. □

Proof of Theorem 1: Now the Theorem 1 can be proved, based on lemmas 1-3. Let $\pi = b_1, \ldots, b_l$ be a derivation for $T \cup T_E \vdash_E a$, where $b_l \sim a$. We can assume that the a and all b_i-s are reduced. We need to show that $\delta(\pi)$

is a derivation for $T \cup T_E^C \vdash_E a$. This completes the proof, because $\sigma(\pi)$ is C-exponent-ground by Lemma 1.

Because $b_l \sim a$ and both b_l and a are reduced, by Lemma 2, we have $\delta(b_l) \sim \delta(a)$. By Lemma 1, $\delta(a) = a$ since a is C-exponent-ground, so we have $\delta(b_l) \sim a$. To prove that $\delta(\pi)$ is a derivation for $T \cup T_E^C \vdash_E a$, we only need to show for each $i \in \{1, \ldots, l\}$ that $\delta(b_i)$ can be obtained from $\{\delta(b_1), \ldots, \delta(b_{i-1})\}$ by applying one of the Horn clauses from $T \cup T_E^C$. 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.

1. b_i is obtained by applying some C-exponent-ground clause from T. There exists a clause $a_1, \ldots, a_n \to a_0$ in T such that $a_0, \ldots, a_n$ are C-exponent-ground. There exists a substitution θ such that $a_0\theta = b_i$ and for each $j \in \{1, \ldots, n\}$ there exists $k_j \in \{1, \ldots, n\}$ with $a_j\theta = b_{k_j}$. Since a_j is C-exponent-ground and θ is reduced, $a_j\theta$ is exponent-reduced. By Lemma 2, $\delta(a_j\theta) \sim \delta(b_{k_j})$ for all $j \in \{0, \ldots, n\}$. By Lemma 3, $a_j\delta(\theta) \sim \delta(b_{k_j})$. We can apply the same clause $a_1, \ldots, a_n \to a_0$ in T with the substitution $\delta(\theta)$ to $\delta(b_{k_1}), \ldots, \delta(b_{k_n})$ and obtain $\delta(b_{k_0}) = \delta(b_i)$.

2. b_i is obtained by applying $I(x), I(y) \to I(e(x, y))$. 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 $t \sim e(s, r)$. We need to show that $I(\delta(t))$ can be obtained from $I(\delta(s))$ and $I(\delta(r))$. Since s and r are reduced by induction hypothesis, by Lemma 2 we have $\delta(t) \sim e(\delta(s), \delta(r))$. We apply the clause $I(x), I(y) \to I(e(x, y))$ from theory T_E^C to get $I(\delta(e(s, r))$ from $I(\delta(s))$ and $I(\delta(r))$.

3. b_i is obtained by applying $I(x) \to I(x^{-1})$. 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 2 to obtain $\delta(t) \sim \delta(s^{-1}) = \delta(s)$. Hence, $\delta(b_i) = I(\delta(t)) \sim I(\delta(s)) = \delta(b_j)$.

4. b_i is obtained by applying $I(x), I(y) \to I(x \uparrow y)$. 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 $I(\delta(t))$ can be obtained from $I(\delta(s))$ and $I(\delta(r))$.
   Since s and r are reduced, $s \uparrow r$ is exponent-reduced. By Lemma 2, we have $\delta(t) = \delta(s \uparrow r)$, so it is enough to show that $I(\delta(s \uparrow r))$ can be obtained from $I(\delta(s))$ and $I(\delta(r))$. Consider three subcases:

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

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

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

5. b_i is obtained by applying $I(x), I(y) \rightarrow I(x \star y)$. 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 $t \sim s \star r$. We need to show that $I(\delta(t))$ can be obtained from $I(\delta(s))$ and $I(\delta(r))$, and it can be done in the same way as it was done for exponentiation, using the rules $I(x), I(c) \rightarrow I(x \star c)$ and $I(x), I(c) \rightarrow I(x \star c^{-1})$. □

3.3 Encoding the Terms to Equivalence Classes

Now there should be defined a theory T_C that will include properties of bilinear pairings and support the equivalence relations *mod* E so that instead of using derivation $\vdash_E$ it would be possible to use a simple syntactical derivation $\vdash$. The only equation that will be used even in syntactical derivation is $e(x, y) \sim e(y, x)$. This property does not have to be encoded since it does not make a problem for ProVerif, and the equation will be defined separately.

We also have to describe the predicates that will be used by ProVerif. After constructing the theory, it is necessary to prove that, if the protocol is secure in theory T_C, it is also secure in theory T_E. In other words, it is needed to prove that, for any protocol theory T_P, we have that $T_P \cup T_E \vdash_E a$ iff $T_P \cup T_C \vdash a'$, where a' is the encoding of a. That would mean that the theory T_C can be used for verification of cryptographic protocols that use bilinear pairings with products in exponents.

Let $C = \{c_1, \ldots, c_m\}$ be the set of pure, ground terms used in the derivation. Let Σ be the signature that contains all the constant, functional and predicate symbols that are used in the protocol. Define $\Sigma^{pair} = (\Sigma \setminus \{\uparrow, ^{-1}, \star\}) \cup \{0, succ, prev, exp, mult\}$ as the new signature.

The constant 0 and the unary functions *succ* and *prev* are used for encoding integers, as in [6]. For example, the integer n will be encoded as $succ^n(0) = succ(succ(\ldots succ(0)\ldots))$, and $-n$ as $prev^n(0)$. There are conversion functions $i2t(n)$ (integer to term) and $t2i(t)$ (term to integer), where $t2i(i2t(n)) = n$. These functions are actually not a part of the theory, and they are needed only for its explanation.

Both *mult* and *exp* are functions 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 the term $exp(s, i2t(n_1), \ldots, i2t(n_m))$ over Σ^{pair}. Similarly, a term of the form $s \star c_1^{(n_1)} \star \ldots \star c_m^{(n_m)}$ will be encoded as the term $mult(s, i2t(n_1), \ldots, i2t(n_m))$.

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 $t = prev(t')$, and incr(t) = succ(t) otherwise;

- $decr(t) = t'$, if $t = decr(t')$, and decr(t) = prev(t) otherwise.

For each $i \in \{1, \ldots, m\}$ there must be defined the functions $incr_i$ and $decr_i$, that will be used in increasing powers of multipliers and exponents. We have to define different functions for multiplication and exponentiation in order to distinguish these operations. Let these functions be $incr_i^{exp}$ for exponentiation, and $incr_i^{mult}$ for multiplication. All of them are also metatheoretical, and these notations will not be present in the theory T_C. We define the following:

- $incr_i^{exp}(exp(t_0, \ldots, t_m)) = t_0$ if $t_i = prev(0)$ and $t_j = 0$ for all $j \neq i$;

- $incr_i^{exp}(exp(t_0, \ldots, t_m)) = exp(t_0, \ldots, t_{i-1}, incr(t_i), t_{i+1}, \ldots, t_m)$ otherwise;

- $incr_i^{exp}(t) = incr_i^{exp}(exp(t, 0, \ldots, 0))$ for t not of the form $exp(t_0, \ldots, t_m)$;

- $decr_i^{exp}(exp(t_0, \ldots, t_m)) = t_0$ if $t_i = succ(0)$ and $t_j = 0$ for all $j \neq i$;

- $decr_i^{exp}(exp(t_0, \ldots, t_m)) = exp(t_0, \ldots, t_{i-1}, decr(t_i), t_{i+1}, \ldots, t_m)$ otherwise;

- $decr_i^{exp}(t) = decr_i^{exp}(exp(t, 0, \ldots, 0))$ for t not of the form $exp(t_0, \ldots, t_m)$;

- $incr_i^{mult}(mult(t_0, \ldots, t_m)) = t_0$ if $t_i = prev(0)$ and $t_j = 0$ for all $j \neq i$;

- $incr_i^{mult}(mult(t_0, \ldots, t_m)) = mult(t_0, \ldots, t_{i-1}, incr(t_i), t_{i+1}, \ldots, t_m)$ otherwise;

- $incr_i^{mult}(t) = incr_i^{mult}(mult(t, 0, \ldots, 0))$
for t not of the form $mult(t_0, \ldots, t_m)$;

- $decr_i^{mult}(mult(t_0, \ldots, t_m)) = t_0$ if $t_i = succ(0)$ and $t_j = 0$ for all $j \neq i$;

- $decr_i^{mult}(mult(t_0,\ldots,t_m)) = mult(t_0,\ldots,t_{i-1},decr(t_i),t_{i+1},\ldots,t_m)$ otherwise;
- $decr_i^{mult}(t) = decr_i^{mult}(mult(t,0,\ldots,0))$
  for t not of the form $mult(t_0,\ldots,t_m)$.

Now we need to transform the terms over Σ into terms of Σ^{pair}, so that the terms could be used in ProVerif. For a C-exponent-ground term t over Σ, define recursively its encoding $\ulcorner t \urcorner$, which is a term over Σ^{pair}:

- $\ulcorner x \urcorner = x$ for a variable x;
- $\ulcorner f(t_1,\ldots,t_n) \urcorner = f(\ulcorner t_1 \urcorner,\ldots,\ulcorner t_n \urcorner)$ for $f \notin \{\uparrow, ^{-1}, \star, e\}$;
- $\ulcorner t \uparrow c_i \urcorner = incr_i^{exp}(\ulcorner t \urcorner)$;
- $\ulcorner t \uparrow c_i^{-1} \urcorner = decr_i^{exp}(\ulcorner t \urcorner)$;
- $\ulcorner t \star c_i \urcorner = incr_i^{mult}(\ulcorner t \urcorner)$;
- $\ulcorner t \star c_i^{-1} \urcorner = decr_i^{mult}(\ulcorner t \urcorner)$;
- $\ulcorner e(t_1 \star c_i, t_2) \urcorner = \ulcorner e(t_1,t_2) \uparrow c_i \urcorner$;
- $\ulcorner e(t_1, t_2 \star c_i) \urcorner = \ulcorner e(t_1,t_2) \uparrow c_i \urcorner$;
- $\ulcorner e(t_1,t_2) \urcorner = e(\ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner)$.

Note that, in the given definition of $\ulcorner \cdot \urcorner$, we do not need to define recursively the encoding of bilinear mapping (we do not need to write $\ulcorner e(t_1 \star c_i, t_2) \urcorner = \ulcorner \ulcorner e(t_1,t_2) \urcorner \uparrow c_i \urcorner$), since the encoding function is applied to $e(t_1,t_2)$ later, according to the rule $\ulcorner t \uparrow c_i \urcorner = incr_i^{exp}(\ulcorner t \urcorner)$.

Lemma 4 *For C-exponent-ground terms t and s, if $t \sim s$, then $\ulcorner t \urcorner \sim \ulcorner s \urcorner$.*

Proof: 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. $(x \uparrow y) \uparrow z = (x \uparrow z) \uparrow y$;
   $(x \star y) \star z = (x \star z) \star y$;
   $e(x, y \star z) = e(x,y) \uparrow z$
   can be applied from left to right and from right to left a number of times. The transformation preserves $\ulcorner . \urcorner$ according to the definition of $\ulcorner . \urcorner$.

2. $(x \uparrow y) \uparrow y^{-1} = x$;
   $(x \star y) \star y^{-1} = x$
   can be applied from left to right a number of times. The transformation preserves $\ulcorner . \urcorner$ according to the definition of $\ulcorner . \urcorner$.

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

4. $e(x, y) = e(y, x)$.
   Does not involve any transformations and therefore preserves $\ulcorner . \urcorner$. It is important that we do not achieve this equivalence by applying $\ulcorner . \urcorner$, and this property is not encoded. In ProVerif, we will define this equation in the heading of the protocol. In this theory, we assume that it holds and does not have to be proven by encoding. □

Example: Let $C = \{c_1, c_2, c_3\}$. The expression $e(c_1 \star P, c_2 \star Q)$ is encoded into $exp(e(P, Q), succ(0), succ(0), 0)$. We carry over all the ground-variable multipliers from G_1 into G_T, and therefore the C-exponent-ground terms that are congruent by bilinearity have the same syntactical representation.

3.4 Derivation Rules for Encoded Terms

Now we are going to define the theory T_C that will deal only with encoded terms and their syntactical derivations. It will be similar to the theory T_C that was defined in the [6], but it contains additional rules for bilinear pairings.

The following rules are dealing with integers: the intruder must be able to derive any integer term.

1. $I(0)$;

2. $I(x) \to I(succ(x))$;

3. $I(x) \to I(prev(x))$.

The intruder must be able to switch between t and $exp(t, 0, \ldots, 0)$, between t and $mult(t, 0, \ldots, 0)$.

1. $I(x) \to I(exp(x, 0, \ldots, 0))$;

2. $I(exp(x, 0, \ldots, 0)) \to I(x)$;

3. $I(x) \to I(mult(x, 0, \dots, 0))$;

4. $I(mult(x, 0, \dots, 0)) \to I(x)$.

If the intruder knows c_i, he is allowed to multiply (exponentiate) the term with $c_i^{(n)}$ for some integer n. This kind of reduction works better with ProVerif than just multiplying (exponentiating) a term with c_i n times.

1. $I(c_i), I(y), I(exp(x_0, x_1, \dots, x_m)) \to$
   $I(exp(x_0, \dots, x_{i-1}, y, x_{i+1}, \dots, x_m))$ for each $c_i \in C$;

2. $I(c_i), I(y), I(mult(x_0, x_1, \dots, x_m)) \to$
   $I(mult(x_0, \dots, x_{i-1}, y, x_{i+1}, \dots, x_m))$ for each $c_i \in C$.

In ProVerif, something must be done with the addition of exponents. When we try to encode $e(x \star c^{(x_1)} \star \dots \star c^{(x_m)}, y \star c^{(y_1)} \star \dots \star c^{(y_m)})$, we get something like $e(x, y) \uparrow c^{(z_1)} \uparrow \dots \uparrow c^{(z_m)}$, where for each i: $z_i = x_i + y_i$. We need to define addition of exponents by introducing a predicate A: $A(x, y, z)$ is true iff $z = i2t(t2i(x) + t2i(y))$.

The problem is that we cannot list all the possible variants like in case of multiplication(exponentiation), because we would have an infinite number of clauses. We need to define this predicate recursively. We will use auxiliary predicates $INCR(x, incr(x))$ and $DECR(x, decr(x))$:

1. $INCR(x, incr(x))$ for $x \in \{0, succ(x), prev(x)\}$;

2. $DECR(x, decr(x))$ for $x \in \{0, succ(x), prev(x)\}$;

3. $A(x, 0, x)$;

4. $A(x, y, z)$, $INCR(z, w) \to A(x, s(y), w)$;

5. $A(x, y, z)$, $DECR(z, w) \to A(x, p(y), w)$.

Now define the rule that allows the intruder to use bilinear pairing function: he gets the result of applying bilinear mapping e, transferring all the possible exponents out of G_1 into G_T:

1. $A(x_1, y_1, z_1)$, $\dots$,$A(x_m, y_m, z_m)$,
   $I(mult(x, x_1, \dots, x_m)), \dots, I(mult(y, y_1, \dots, y_m)) \to$
   $I(exp(e(x, y), z_1, \dots, z_m))$.

Define new predicates E,M, and P which will express exponentiation, multiplication and pairing for C-exponent-ground terms:
$E(x, y, z)$ is true iff $x \uparrow y \sim z$.
$M(x, y, z)$ is true iff $x \star y \sim z$.
$P(x, y, z)$ is true iff $e(x, y) \sim z$.

1. $E(t, c_i, incr_i^{exp}(t))$ for each $c_i \in C$ and $t \in A_i^+$;
2. $E(t, c_i, decr_i^{exp}(t))$ for each $c_i \in C^{-1}$ and $t \in A_i^-$;
   - $A_i^+ = \{x, exp(x_0, \ldots, x_m), exp(x_0, \ldots, prev(x_i), \ldots, x_m), \ulcorner x \uparrow c_i^{-1} \urcorner\}$;
   - $A_i^- = \{x, exp(x_0, \ldots, x_m), exp(x_0, \ldots, succ(x_i), \ldots, x_m), \ulcorner x \uparrow c_i \urcorner\}$;
3. $M(t, c_i, incr_i^{mult}(t))$ for each $c_i \in C$ and $t \in B_i^+$;
4. $M(t, c_i, decr_i^{mult}(t))$ for each $c_i \in C^{-1}$ and $t \in B_i^-$;
   - $B_i^+ = \{x, mult(x_0, \ldots, x_m), mult(x_0, \ldots, prev(x_i), \ldots, x_m), \ulcorner x \star c_i^{-1} \urcorner\}$;
   - $B_i^- = \{x, mult(x_0, \ldots, x_m), mult(x_0, \ldots, succ(x_i), \ldots, x_m), \ulcorner x \star c_i \urcorner\}$.

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.

1. $P(x, y, e(x, y))$;
2. $P(mult(x, x_1, \ldots, x_m), y, exp(e(x, y), x_1, \ldots, x_m))$;
3. $P(x, mult(y, y_1, \ldots, y_m), exp(e(x, y), y_1, \ldots, y_m))$.

   The case $x = mult(x, x_1, \ldots, x_m)$ and $y = mult(y, y_1, \ldots, y_m)$ would have infinite number of clauses. We need to describe this case in another way:
4. $A(x_1, y_1, z_1), \ldots, A(x_m, y_m, z_m) \rightarrow$
   $P(mult(x, x_1, \ldots, x_m), mult(y, y_1, \ldots, y_m), exp(e(x, y), z_1, \ldots, z_m))$.

   This rule may give us answers like $exp(e(x, y), 0, \ldots, 0)$. The intruder may easily transfom these into $e(x, y)$, but there would be some contradictions within the protocol rules defined in a protocol theory T. For example, there would be no congruence between $e(x, y) \uparrow c_1^{-1} \uparrow c_1 = e(x, y)$ and $e(x \star c_1, y \star c_1^{-1}) = exp(e(x, y), 0)$. We need to define a separate rule for this case:
5. $A(x_1, y_1, 0), \ldots, A(x_m, y_m, 0) \rightarrow$
   $P(mult(x, x_1, \ldots, x_m), mult(y, y_1, \ldots, y_m), e(x, y))$.

These predicates allow to encode the rules of any theory T into the rules of theory T_C. In this way, any protocol using bilinear pairings with products in exponents can be encoded to the form appropriate for ProVerif.

Given clauses $r_1, \ldots, r_n \rightarrow r_0$ from a theory T, it is necessary to substitute all non-ground, non-standard subterms with their C-exponent-ground encodings.

1. $M(\ulcorner\theta(s'_1)\urcorner, b_1, x_1)$, $\dots$,$M(\ulcorner\theta(s'_j)\urcorner, b_j, x_j)$,
   $E(\ulcorner\theta(t'_1)\urcorner, d_1, y_1)$, $\dots$,$E(\ulcorner\theta(t'_k)\urcorner, d_k, y_k)$,
   $P(\ulcorner\theta(u'_1)\urcorner, \ulcorner\theta(v'_1)\urcorner, z_1)$, $\dots$,$P(\ulcorner\theta(u'_l)\urcorner, \ulcorner\theta(v'_1)\urcorner, z_l)$,
   $\ulcorner\theta(r_1)\urcorner, \dots, \ulcorner\theta(r_n)\urcorner \to \ulcorner\theta(r_0)\urcorner$

   for each clause $r_1, \dots, r_n \to r_0$ in T where $s_1 = s'_1 \star b_1, \dots, s_j = s'_j \star b_j$, $t_1 = t'_1 \uparrow d_1, \dots, t_k = t'_k \uparrow d_k$, $w_1 = e(u'_1, v'_1), \dots, w_l = e(u'_l, v'_l)$ are non-ground, non-standard sumterms of $r_0, \dots, r_n$, and θ replaces each s_i with a new variable x_i, t_i with y_i, w_i with z_i.

Lemma 5 *Let t and s be ground terms, $c \in C$ and assume that t and s are C-exponent-ground. Then:*

- *$E(\ulcorner t\urcorner, c, \ulcorner t \uparrow c\urcorner)$ is an instance of $E(t, c_i, incr_i^{exp}(t))$.*
- *$E(\ulcorner t\urcorner, c^{-1}, \ulcorner t \uparrow c^{-1}\urcorner)$ is an instance of $E(t, c_i, decr_i^{exp}(t))$.*
- *$M(\ulcorner t\urcorner, c, \ulcorner t \star c\urcorner)$ is an instance of $M(t, c_i, incr_i^{mult}(t))$.*
- *$M(\ulcorner t\urcorner, c^{-1}, \ulcorner t \star c^{-1}\urcorner)$ is an instance of $M(t, c_i, decr_i^{mult}(t))$.*
- *$P(\ulcorner t\urcorner, \ulcorner s\urcorner, \ulcorner e(t, s)\urcorner)$ is an instance of $P(x, y, e(x, y))$.*
- *$P(\ulcorner t \star c_{i1} \star \dots \star c_{ik}\urcorner, \ulcorner s\urcorner, \ulcorner e(t, s) \uparrow c_{i1} \uparrow \dots \uparrow c_{ik}\urcorner)$ is an instance of $P(mult(x, x_1, \dots, x_m), y, exp(e(x, y), x_1, \dots, x_m))$.*
- *$P(\ulcorner t\urcorner, \ulcorner s \star c_{i1} \star \dots \star c_{ik}\urcorner, \ulcorner e(t, s) \uparrow c_{i1} \uparrow \dots \uparrow c_{ik}\urcorner)$ is an instance of $P(mult(x, x_1, \dots, x_m), y, exp(e(x, y), x_1, \dots, x_m))$.*
- *$P(\ulcorner t \star c_{i1} \star \dots \star c_{ik}\urcorner, \ulcorner s \star c_{j1} \star \dots \star c_{jl}\urcorner, \ulcorner e(t, s) \uparrow c_{i1} \uparrow \dots \uparrow c_{ik} \uparrow c_{j1} \uparrow \dots \uparrow c_{jl}\urcorner)$ is an instance of $P(mult(x, x_1, \dots, x_m), mult(y, y_1, \dots, y_m), exp(e(x, y), z_1, \dots, z_m))$, where the variables satisfy the predicates $A(x_1, y_1, z_1), \dots, A(x_m, y_m, z_m)$.*
- *$P(\ulcorner t \star c_{i1} \star \dots \star c_{ik}\urcorner, \ulcorner s \star c_{j1} \star \dots \star c_{jl}\urcorner, \ulcorner e(t, s)\urcorner)$ is an instance of $P(mult(x, x_1, \dots, x_m), mult(y, y_1, \dots, y_m), e(x, y))$, where the variables satisfy the predicates $A(x_1, y_1, 0), \dots, A(x_m, y_m, 0)$.*

Proof: The proof of this lemma is based on the definition of the encoding function $\ulcorner\cdot\urcorner$. In the rules, we may substitute the variables with any terms.

- $E(\ulcorner t\urcorner, c, \ulcorner t \uparrow c\urcorner) = E(\ulcorner t\urcorner, c_i, incr_i^{exp}(\ulcorner t\urcorner)))$, which is an instance of $E(t, c_i, incr_i^{exp}(t))$ for some i where $c = c_i$.

- $E(\ulcorner t \urcorner, c^{-1}, \ulcorner t \uparrow c^{-1} \urcorner) = E(\ulcorner t \urcorner, c_i, decr_i^{exp}(\ulcorner t \urcorner)))$, which is an instance of $E(t, c_i, decr_i^{exp}(t))$ for some i where $c = c_i$.

- $M(\ulcorner t \urcorner, c, \ulcorner t \star c \urcorner) = M(\ulcorner t \urcorner, c_i, incr_i^{mult}(\ulcorner t \urcorner)))$, which is an instance of $M(t, c_i, incr_i^{mult}(t))$ for some i where $c = c_i$.

- $M(\ulcorner t \urcorner, c^{-1}, \ulcorner t \star c^{-1} \urcorner) = M(\ulcorner t \urcorner, c_i, decr_i^{mult}(\ulcorner t \urcorner)))$, which is an instance of $M(t, c_i, decr_i^{mult}(t))$ for some i where $c = c_i$.

- $P(\ulcorner t \urcorner, \ulcorner s \urcorner, \ulcorner e(t,s) \urcorner) = P(\ulcorner t \urcorner, \ulcorner s \urcorner, e(\ulcorner s \urcorner, \ulcorner t \urcorner))$, which is an instance of $P(x, y, e(x,y))$.

- $P(\ulcorner t \star c_{i1} \star \ldots \star c_{ik} \urcorner, \ulcorner s \urcorner, \ulcorner e(t,s) \uparrow c_{i1} \uparrow \ldots \uparrow c_{ik} \urcorner) =$
  $P(incr_{ik}^{mult}(\ldots(incr_{i1}^{mult}(\ulcorner t \urcorner))\ldots), \ulcorner s \urcorner,$
  $incr_{ik}^{exp}(\ldots(incr_{i1}^{exp}(e(\ulcorner t \urcorner, \ulcorner s \urcorner)))\ldots)),$
  which is an instance of $P(mult(x, x_1, \ldots, x_m), y, exp(e(x,y), x_1, \ldots, x_m))$, according to the definition of $incr^{exp}$ and $incr^{mult}$.

- $P(\ulcorner t \urcorner, \ulcorner s \star c_{i1} \star \ldots \star c_{ik} \urcorner, \ulcorner e(t,s) \uparrow c_{i1} \uparrow \ldots \uparrow c_{ik} \urcorner) =$
  $P(incr_{ik}^{mult}(\ldots(incr_{i1}^{mult}(\ulcorner t \urcorner))\ldots), incr_{jl}^{mult}(\ldots(incr_{j1}^{mult}(\ulcorner s \urcorner))\ldots),$
  $incr_{jl}^{exp}(\ldots(incr_{j1}^{exp}(\ldots(incr_{ik}^{exp}(\ldots(incr_{i1}^{exp}(e(\ulcorner t \urcorner, \ulcorner s \urcorner)))\ldots))\ldots))\ldots).$
  This is an instance of
  $P(mult(x, x_1, \ldots, x_m), mult(y, y_1, \ldots, y_m), exp(e(x,y), z_1, \ldots, z_m))$,
  where the predicates $A(x_1, y_1, z_1), \ldots, A(x_m, y_m, z_m)$ are true, according to the definitions of the functions $incr^{exp}$, $incr^{mult}$, and of the predicate $A(x,y,z)$.

- $P(\ulcorner t \urcorner, \ulcorner s \star c_{i1} \star \ldots \star c_{ik} \urcorner, \ulcorner e(t,s) \urcorner) =$
  $P(incr_{ik}^{mult}(\ldots(incr_{i1}^{mult}(\ulcorner t \urcorner))\ldots), incr_{jl}^{mult}(\ldots(incr_{j1}^{mult}(\ulcorner s \urcorner))\ldots),$
  $e(\ulcorner t \urcorner, \ulcorner s \urcorner)).$
  This is an instance of $P(mult(x, x_1, \ldots, x_m), mult(y, y_1, \ldots, y_m), e(x,y))$, where the predicates $A(x_1, y_1, 0), \ldots, A(x_m, y_m, 0)$ are true, according to the definitions of the functions $incr^{exp}$, $incr^{mult}$, and of the predicate $A(x,y,z)$.

Example: Suppose that $C = \{a, b\}$, and we are given $\ulcorner t \star a \star b^{-1} \urcorner = mult(t, succ(0), prev(0))$ and $\ulcorner s \star b \urcorner = mult(s, 0, prev(0))$. According to the definition of P, $P(\ulcorner t \star a \star b^{-1} \urcorner, \ulcorner s \star b \urcorner, \ulcorner e(t \star a \star b^{-1}, s \star b) \urcorner)$ is the instance of $P(mult(t, succ(0), prev(0)), mult(t, 0, succ(0)), exp(e(t,s), succ(0), 0))$, where $A(succ(0), 0, succ(0)), A(prev(0), succ(0), 0)$ is true. On the other hand, using the function $\ulcorner . \urcorner$ gives us the same value $exp(e(t,s), succ(0), 0))$:
$\ulcorner e(t \star a \star b^{-1}, s \star b) \urcorner$
$= \ulcorner e(t \star a, s \star b) \uparrow b^{-1} \urcorner$

$= decr_b^{exp}(\ulcorner e(t \star a, s \star b)\urcorner)$

$= decr_b^{exp}(\ulcorner e(t, s \star b) \uparrow a\urcorner)$

$= decr_b^{exp}(incr_a^{exp}(\ulcorner e(t, s \star b)\urcorner))$

$= decr_b^{exp}(incr_a^{exp}(\ulcorner e(t, s) \uparrow b\urcorner))$

$= decr_b^{exp}(incr_a^{exp}(incr_b^{exp}(\ulcorner e(t, s)\urcorner)))$

$= decr_b^{exp}(incr_a^{exp}(incr_b^{exp}(e(\ulcorner t\urcorner, \ulcorner s\urcorner))))$

$= decr_b^{exp}(incr_a^{exp}(incr_b^{exp}(e(t, s))))$

$= decr_b^{exp}(incr_a^{exp}(exp(e(t, s), 0, succ(0))))$

$= decr_b^{exp}(exp(e(t, s), succ(0), succ(0)))$

$= exp(e(t, s), succ(0), 0).$

Now let us consider the clauses given by the rule:

$M(\ulcorner\theta(s'_1)\urcorner, b_1, x_1), \ldots, M(\ulcorner\theta(s'_j)\urcorner, b_j, x_j),$

$E(\ulcorner\theta(t'_1)\urcorner, d_1, y_1), \ldots, E(\ulcorner\theta(t'_k)\urcorner, d_k, y_k),$

$P(\ulcorner\theta(u'_1)\urcorner, \ulcorner\theta(v'_1)\urcorner, z_1), \ldots, P(\ulcorner\theta(u'_l)\urcorner, \ulcorner\theta(v'_1)\urcorner, z_l),$

$\ulcorner\theta(r_1)\urcorner, \ldots, \ulcorner\theta(r_n)\urcorner \rightarrow \ulcorner\theta(r_0)\urcorner$

for some clause $A = (r_1, \ldots, r_n \rightarrow r_0)$.

Let us denote the clause in T_C resulting from A by A^*. First, if A does not contain neither multiplication(exponentiation) symbols nor the bilinear pairing function, then $A^* = A$. If A contains some term $t \uparrow d$ with a non-ground term t, it is replaced by a fresh variable y, and the relation between t, d, and y is captured by adding $E(t, d, y)$ to the clause. Similarly, a term $t \star d$ adds a new clause $M(t, d, y)$, and $e(s, t)$ adds a new clause $P(s, t, y)$. These steps are applied recursively to the remaining non-ground, non-standard subterms of A, including the subterms of s and t. All terms are encoded using $\ulcorner . \urcorner$ to obtain terms over Σ^{pair}.

Theorem 2 *Let T be a non-trivial, C-exponent-ground theory over Σ and $b = p(t)$ be a C-exponent-ground atom over Σ, with p being a predicate occuring in T. Then, $T \cup T_E \vdash_E b$ iff $T_C \vdash \ulcorner b\urcorner$.*

If we prove this theorem, it means that any derivation *mod* E(using the properties of bilinear mappings) can be reduced to a purely syntactical derivation and can be analyzed by ProVerif.

Lemma 6 *If there exists a C-exponent-ground derivation for $T \cup T_E^C \vdash_E b$ obtained using C-exponent-ground substitutions, then $T_C \vdash \ulcorner b\urcorner$.*

Proof: Let $\pi = b_1, \ldots, b_l$ be a C-exponent-ground derivation for $T \cup T_E^C \vdash_E b$ obtained using C-exponent-ground substitutions. Prove by induction on length of π:

- Base: If $l = 0$, there is no derivation

- Step: Let $\pi_{<l} = b_1, \ldots, b_{l-1}$. We know that $b \sim b_l$ can be derived from $\pi_{<l}$ by applying a clause from $T \cup T_E^C$ using a C-exponent-ground substitution σ. It is enough to show that $\ulcorner b \urcorner$ can be syntactically derived from $\ulcorner \pi_{<l} \urcorner$ using T_C. There are two cases to consider:

1. If b is obtained using a clause of T_E^C, then $b = I(t)$ for some C-exponent-ground term t. There are three subcases:

   (a) The set $\pi_{<l}$ contains atoms $I(r)$ for a C-exponent-ground r and $I(c_i)$ for $c_i \in C$, such that $t \sim r \uparrow c_i$ or $t \sim r \uparrow c_i^{-1}$. The atom $I(\ulcorner t \urcorner)$ can be obtained from $I(\ulcorner r \urcorner)$ and $I(\ulcorner c_i \urcorner)$ using the following clauses:

      i. $I(x) \to I(exp(x, 0, \ldots, 0))$, if the reduced form of r is standard.

      ii. $I(c_i), I(y), I(exp(x_0, x_1, \ldots, x_m)) \to$
      $I(exp(x_0, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_m))$ is used with an appropriate integer term derived by integer-derivation clauses:
      $I(0)$; $I(x) \to I(succ(x))$; $I(x) \to I(prev(x))$.

      iii. If the reduced form of t is standard, then $I(exp(x, 0, \ldots, 0)) \to I(x)$ is applied.

   (b) The set $\pi_{<l}$ contains atoms $I(r)$ for a C-exponent-ground r and $I(c_i)$ for $c_i \in C$, such that $t \sim r \star c_i$ or $t \sim r \star c_i^{-1}$. The atom $I(\ulcorner t \urcorner)$ can be obtained from $I(\ulcorner r \urcorner)$ and $I(\ulcorner c_i \urcorner)$ using analogical clauses that are defined for multiplication in T_C.

   (c) The set $\pi_{<l}$ contains atoms $I(r)$ and $I(s)$ for C-exponent-ground r and s, such that $t \sim e(r, s)$. The atom $I(\ulcorner t \urcorner)$ can be obtained from $I(\ulcorner r \urcorner)$ and $I(\ulcorner s \urcorner)$ using the following clauses:

      i. If r(or s) is not of the form $mult(x_0, x_1, \ldots, x_m)$, then $I(x) \to I(mult(x, 0, \ldots, 0))$ must be applied to r(or s) in order to get $I(mult(r_0, r_1, \ldots, r_m))$ and $I(mult(s_0, s_1, \ldots, s_m))$, where $r \sim mult(r_0, r_1, \ldots, r_m)$ and $s \sim mult(s_0, s_1, \ldots, s_m)$.

      ii. Apply the rule
      $A(x_1, y_1, z_1), \ldots, A(x_m, y_m, z_m)$,
      $I(mult(x, x_1, \ldots, x_m)), \ldots, I(mult(y, y_1, \ldots, y_m)) \to$
      $I(exp(e(x, y), z_1, \ldots, z_m))$
      to $mult(r_0, r_1, \ldots, r_m)$ and $mult(s_0, s_1, \ldots, s_m)$.

2. If b is obtained by some C-exponent-ground clause $r_1, \ldots, r_n \to r_0$ of T, there exists a C-exponent-ground substitution σ such that $b \sim \sigma(r_0)$ and all $\sigma(r_1), \ldots, \sigma(r_n)$ belong to $\pi_{<l}$ $(mod\, E)$. The $\ulcorner b \urcorner$ can be obtained

by using the clause that uses predicates E,M, and P. Denote the clause $r_1,\dots,r_n \to r_0$ as $R \to S$. We will use the same variables that are defined in the statement of this rule,
$M(\ulcorner\theta(s'_1)\urcorner, b_1, x_1),\dots,M(\ulcorner\theta(s'_j)\urcorner, b_j, x_j)$,
$E(\ulcorner\theta(t'_1)\urcorner, d_1, y_1),\dots,E(\ulcorner\theta(t'_k)\urcorner, d_k, y_k)$,
$P(\ulcorner\theta(u'_1)\urcorner,\ulcorner\theta(v'_1)\urcorner, z_1),\dots,P(\ulcorner\theta(u'_l)\urcorner,\ulcorner\theta(v'_1)\urcorner, z_l)$,
$\ulcorner\theta(r_1)\urcorner,\dots,\ulcorner\theta(r_n)\urcorner \to \ulcorner\theta(r_0)\urcorner$.
Define a substitution σ^*, which will be applied to $R \to S$ to obtain $\ulcorner b\urcorner$ as follows:

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

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

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

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

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

   i. if $u \in \{s_1,\dots,s_k\}$, then $\theta(u) = x_i$ for some i.
   ii. if $u \in \{t_1,\dots,t_k\}$, then $\theta(u) = y_i$ for some i.
   iii. if $u \in \{w_1,\dots,w_k\}$, then $\theta(u) = z_i$ for some i.

   The claim follows from the definition of σ^*.

Now we have that $\sigma^*(\ulcorner\theta(r_i)\urcorner) = \ulcorner\sigma(r_i)\urcorner$ for each $i \in \{0,\dots,n\}$; in particular, $\sigma^*(\ulcorner\theta(r_i)\urcorner) \in \ulcorner\pi_{<l}\urcorner$ for each $i \in \{1,\dots,n\}$, and we obtain $\sigma^*(\ulcorner\theta(r_i)\urcorner) = \ulcorner\sigma(r_0)\urcorner = \ulcorner b\urcorner$ by applying $R \to S$ with substitution σ^* (the equality follows from Lemma 4). It remains to prove that:
$\sigma^*(M(\ulcorner\theta(s'_i)\urcorner, b_i, x_i))$,
$\sigma^*(E(\ulcorner\theta(t'_i)\urcorner, d_i, y_i))$,
$\sigma^*(P(\ulcorner\theta(u'_i)\urcorner,\ulcorner\theta(v'_i)\urcorner, z_i))$
can all be derived from T_C.

- We have $\sigma^*(E(\ulcorner\theta(t'_i)\urcorner, d_i, y_i)) = E(\sigma^*(\ulcorner\theta(t'_i)\urcorner), d_i, \sigma^*(y_i))$,
  which is equal to $E(\ulcorner\sigma(t'_i)\urcorner, d_i, \ulcorner\sigma(t_i)\urcorner)$,
  and therefore is equal to $E(\ulcorner\sigma(t'_i)\urcorner, d_i, \ulcorner\sigma(t'_i) \uparrow d_i\urcorner)$.

By Lemma 5, this fact is an instance of $E(t, c_i, incr_i^{exp}(t))$ or $E(t, c_i, decr_i^{exp}(t))$, depending on whether d_i belongs to C or C^{-1}.

- Analogically, we have
$\sigma^*(M(\ulcorner\theta(s'_i)\urcorner, b_i, x_i))$
$= M(\sigma^*(\ulcorner\theta(s'_i)\urcorner), b_i, \sigma^*(x_i))$
$= M(\ulcorner\sigma(s'_i)\urcorner, b_i, \ulcorner\sigma(s_i)\urcorner)$
$= M(\ulcorner\sigma(s'_i)\urcorner, b_i, \ulcorner\sigma(s'_i) \star b_i\urcorner)$.
By Lemma 5, this fact is an instance of $M(t, c_i, incr_i^{mult}(t))$ or $M(t, c_i, decr_i^{mult}(t))$, depending on whether b_i belongs to C or C^{-1}.
- We have
$\sigma^*(P(\ulcorner\theta(u'_i)\urcorner, \ulcorner\theta(v'_i)\urcorner, z_i)) = P(\sigma^*(\ulcorner\theta(u'_i)\urcorner), \sigma^*(\ulcorner\theta(v'_i)\urcorner), \sigma^*(z_i))$, which is equal to $P(\ulcorner\sigma(u'_i)\urcorner, \ulcorner\sigma(v'_i)\urcorner, \ulcorner\sigma(w_i)\urcorner)$, and therefore to $P(\ulcorner\sigma(u'_i)\urcorner, \ulcorner\sigma(v'_i)\urcorner, \ulcorner e(\sigma(u'_i), \sigma(v'_i))\urcorner)$. By Lemma 5, this fact is an instance of:
  - $P(x, y, e(x, y))$, if the head symbol of both u'_i and v'_i is not $\star$.
  - $P(mult(x, x_1, \ldots, x_m), y, exp(e(x, y), x_1, \ldots, x_m))$, if the head symbol of u'_i is $\star$, and the head symbol of v'_i is not $\star$.
  - $P(mult(x, x_1, \ldots, x_m), y, exp(e(x, y), x_1, \ldots, x_m))$, if the head symbol of u'_i is not $\star$, and the head symbol of v'_i is $\star$.
  - $P(mult(x, x_1, \ldots, x_m), mult(y, y_1, \ldots, y_m),$
    $exp(e(x, y), z_1, \ldots, z_m))$,
    (where the variables satisfy $A(x_1, y_1, z_1), \ldots, A(x_m, y_m, z_m)$), if the head symbols of both u'_i and v'_i are $\star$ and z_i is of the form $exp(\ldots)$.
  - $P(mult(x, x_1, \ldots, x_m), mult(y, y_1, \ldots, y_m), e(x, y))$, (where the variables satisfy the predicates $A(x_1, y_1, 0), \ldots, A(x_m, y_m, 0)$), if the head symbols of both u'_i and v'_i are $\star$ and z_i is not of the form $exp(\ldots)$. □

3.5 Decoding the Terms: Soundness of the Reduction

Now we are going to prove that $T_C \vdash \ulcorner b\urcorner$ implies $T \cup T_E \vdash_E b$. 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 Σ^{pair} back into terms over Σ.

In the process of decoding, we will use non-triviality of the protocol theory T: that there exists some u such that $T \cup T_E \vdash_E I(u)$. If T was empty, we would not need to test it with ProVerif.

First, we need to extend the domain of the function $t2i$, that was defined before:

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

Define the decoding function, a mapping $\llcorner\cdot\lrcorner$ from terms over Σ^{pair} to terms over Σ:

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

There must be relationship between the functions $\ulcorner . \urcorner$ and $\llcorner . \lrcorner$. Everything that was encoded may be later decoded. These functions are not the inverse functions of each other, since syntactically different congruent terms have the same encoding, an the decoding function does not know the initial values of an encoded term. But we do not need the syntactical equivalence of a term t and $\llcorner \ulcorner t \urcorner \lrcorner$, it would be enough if these were equivalent *mod E*.

Lemma 7 *Let t be a C-exponent-ground term over Σ. Then $\llcorner \ulcorner t \urcorner \lrcorner \sim t$.*

Proof: This lemma can be proven by structural induction on 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 three cases for a non-standard form:

- 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 $\ulcorner t' \urcorner = exp(\ulcorner t_0 \urcorner, i2t(k_1), \ldots, i2t(k_m))$. By definition of $\llcorner\cdot\lrcorner$ and the fact that $t2i(i2t(k)) = k$, we obtain $\llcorner \ulcorner t' \urcorner \lrcorner = \llcorner \ulcorner t_0 \urcorner \lrcorner \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$. The induction hypothesis yields that $\llcorner \ulcorner t_0 \urcorner \lrcorner \sim t_0$, and therefore $\llcorner \ulcorner t' \urcorner \lrcorner \sim t_0 \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$. Hence, $\llcorner \ulcorner t' \urcorner \lrcorner \sim t'$. Since $t \sim t'$, Lemma 4 implies that $\ulcorner t' \urcorner = \ulcorner t \urcorner$ and so $\llcorner \ulcorner t' \urcorner \lrcorner = \llcorner \ulcorner t \urcorner \lrcorner$. Consequently, $t \sim t' \sim \llcorner \ulcorner t \urcorner \lrcorner$.

- If $t' = t_0 \star c_1^{(k_1)} \star \ldots \star c_m^{(k_m)}$ for some integers $k_1, \ldots, k_m$ and a C-exponent ground term t_0, then $\ulcorner t' \urcorner = mult(\ulcorner t_0 \urcorner, i2t(k_1), \ldots, i2t(k_m))$. As in the case of exponentiation, we get $\llcorner \ulcorner t' \urcorner \lrcorner = \llcorner \ulcorner t_0 \urcorner \lrcorner \star c_1^{(k_1)} \star \ldots \star c_m^{(k_m)}$. According to induction hypothesis, $\llcorner \ulcorner t' \urcorner \lrcorner \sim t_0 \star c_1^{(k_1)} \star \ldots \star c_m^{(k_m)}$. Applying Lemma 4, we again get that $t \sim t' \sim \llcorner \ulcorner t \urcorner \lrcorner$.

- If $t' = e(r', s')$, then we have three different cases:

  - If r' and s' are standard, then t' is also standard.
  - If either $r' = r_0 \star c_1^{(k_1)} \star \ldots \star c_m^{(k_m)}$ or $s' = s_0 \star c_1^{(k_1)} \star \ldots \star c_m^{(k_m)}$, then
    $\ulcorner t' \urcorner = exp(e(\ulcorner r_0 \urcorner, \ulcorner s' \urcorner), i2t(k_1), \ldots, i2t(k_m))$ or
    $\ulcorner t' \urcorner = exp(e(\ulcorner r' \urcorner, \ulcorner s_0 \urcorner), i2t(k_1), \ldots, i2t(k_m))$.
    By definition of $\llcorner \cdot \lrcorner$, we obtain:
    $\llcorner \ulcorner t' \urcorner \lrcorner = e(\llcorner \ulcorner r' \urcorner \lrcorner, \llcorner \ulcorner s_0 \urcorner \lrcorner) \uparrow c_1^{(k1)} \uparrow \ldots \uparrow c_m^{(km)}$ or
    $\llcorner \ulcorner t' \urcorner \lrcorner = e(\llcorner \ulcorner r_0 \urcorner \lrcorner, \llcorner \ulcorner s' \urcorner \lrcorner) \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$.
    The induction hypothesis yields that $\llcorner \ulcorner r_0 \urcorner \lrcorner \sim r_0$, $\llcorner \ulcorner s_0 \urcorner \lrcorner \sim s_0$, $\llcorner \ulcorner r' \urcorner \lrcorner \sim r'$, and $\llcorner \ulcorner s' \urcorner \lrcorner \sim s'$. We get that:
    $\llcorner \ulcorner t' \urcorner \lrcorner = e(r', s_0) \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$ or
    $\llcorner \ulcorner t' \urcorner \lrcorner = e(r_0, s') \uparrow c_1^{(k_1)} \uparrow \ldots \uparrow c_m^{(k_m)}$.
    We get that $\llcorner \ulcorner t' \urcorner \lrcorner \sim t'$, and, according to Lemma 4, $t \sim \llcorner \ulcorner t' \urcorner \lrcorner$.
  - If both $r' = r_0 \star c_1^{(k_1)} \star \ldots \star c_m^{(k_m)}$ and $s' = s_0 \star c_1^{(l_1)} \star \ldots \star c_m^{(l_m)}$, then
    $\ulcorner t' \urcorner = exp(e(\ulcorner r_0 \urcorner, \ulcorner s_0 \urcorner), i2t(k_1 + l_1), \ldots, i2t(k_m + l_m))$.
    Again, by definition of $\llcorner \cdot \lrcorner$, we get that:
    $\llcorner \ulcorner t' \urcorner \lrcorner = e(\llcorner \ulcorner r_0 \urcorner \lrcorner, \llcorner \ulcorner s_0 \urcorner \lrcorner) \uparrow c_1^{(k1+l1)} \uparrow \ldots \uparrow c_m^{(km+lm)}$.
    By induction hypothesis, $\llcorner \ulcorner t' \urcorner \lrcorner = e(r_0, s_0) \uparrow c_1^{(k1+l1)} \uparrow \ldots \uparrow c_m^{(km+lm)}$. Applying Lemma 4, we again have $t \sim \llcorner \ulcorner t' \urcorner \lrcorner$. □

Lemma 8 *Let t,d, and s be ground terms over Σ^{pair}. If $E(t, d, s)$ can be derived from T_C, then $d \in C \cup C^{-1}$ and $\llcorner s \lrcorner \sim \llcorner t \lrcorner \uparrow d$.*

Proof: 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, $\llcorner s \lrcorner = \llcorner incr_i^{exp}(t) \lrcorner$. We have three different cases for t:

  - Let t be of the form $exp(t_0, \ldots, t_m)$, where $t_i = prev(0)$ and $\forall j \neq 0$ $t_j = 0$. We have that $\llcorner incr_i^{exp}(t) \lrcorner = \llcorner t_0 \lrcorner$. On the other hand, $\llcorner t \lrcorner \uparrow d = \llcorner t \lrcorner \uparrow c_i = \llcorner exp(t_0, \ldots, t_m) \lrcorner \uparrow c_i = \llcorner t_0 \lrcorner \uparrow c_1^{(t2i(t_1))} \uparrow \ldots \uparrow$

$c_m^{(t2i(t_m))} \uparrow c_i$. Since $t_i = prev(0)$ and $\forall j \neq 0 \ t_j = 0$, this equals to $\llcorner t_0 \lrcorner \uparrow c_1^{(0)} \uparrow \ldots \uparrow c_i^{-1} \uparrow \ldots \uparrow c_m^{(0)} \uparrow c_i \sim \llcorner t_0 \lrcorner \uparrow c_i^{(-1)} \uparrow c_i \sim \llcorner t_0 \lrcorner$.

  - Let t be of the form $exp(t_0, \ldots, t_m)$, where either $t_i \neq prev(0)$ or $\exists j \neq i \ t_j \neq 0$. We have that
    $\llcorner incr_i^{exp}(t) \lrcorner$
    $= \llcorner exp(t_0, \ldots, t_{i-1}, incr(t_i), t_{i+1}, \ldots, t_m) \lrcorner$
    $= \llcorner t_0 \lrcorner \uparrow c_1^{(t2i(t_1))} \uparrow \ldots \uparrow c_i^{(t2i(incr(t_i)))} \uparrow \ldots \uparrow c_m^{(t2i(t_m))}$.
    On the other hand,
    $\llcorner t \lrcorner \uparrow d$
    $= \llcorner t \lrcorner \uparrow c_i$
    $= \llcorner exp(t_0, t_1, \ldots, t_m) \lrcorner \uparrow c_i$
    $= \llcorner t_0 \lrcorner \uparrow c_1^{(t2i(t_1))} \uparrow \ldots \uparrow c_m^{(t2i(t_m))} \uparrow c_i$
    $\sim \llcorner t_0 \lrcorner \uparrow c_1^{(t2i(t_1))} \uparrow \ldots \uparrow c_i^{(t2i(t_i)+1)} \uparrow \ldots \uparrow c_m^{(t2i(t_m))}$
    $= \llcorner t_0 \lrcorner \uparrow c_1^{(t2i(t_1))} \uparrow \ldots \uparrow c_i^{(t2i(succ(t_i)))} \uparrow \ldots \uparrow c_m^{(t2i(t_m))}$
    $= \llcorner t_0 \lrcorner \uparrow c_1^{(t2i(t_1))} \uparrow \ldots \uparrow c_i^{(t2i(incr(t_i)))} \uparrow \ldots \uparrow c_m^{(t2i(t_m))}$.
  - If t be not of the form $exp(t_0, \ldots, t_m)$, then, by definition of function $incr_i^{exp}$, we have
    $\llcorner incr_i^{exp}(t) \lrcorner$
    $= \llcorner incr_i^{exp}(exp(t, 0, \ldots, 0)) \lrcorner$
    $= \llcorner exp(t, 0, \ldots, incr_i^{exp}(0), \ldots, 0) \lrcorner$
    $= \llcorner t \lrcorner \uparrow c_1^{(t2i(0))} \uparrow \ldots \uparrow c_i^{(t2i(succ(0)))} \uparrow \ldots \uparrow c_m^{(t2i(0))}$
    $\sim \llcorner t \lrcorner \uparrow c_i$.

- $E(t, d, s) = E(t, c_i, decr_i^{exp}(t))$. The proof is very similar to the case $E(t, d, s) = E(t, c_i, incr_i^{exp}(t))$. We only need to replace all instances of $incr_i^{exp}$ with $decr_i^{exp}$ and use the properties of the function $decr_i^{exp}$. □

Example: Let $C = \{a, b\}$. $E(exp(x_0, x_1, x_2), a^{-1}, exp(x_0, prev(x_1), x_2))$ is a fact of T_C. Consider substitution $\sigma = \{g/x_0, g/x_1, 0/x_2\}$ for a constant g. Then, $E(exp(g, g, 0), a^{-1}, exp(g, prev(g), 0))$ is an instance of T_C. We have that $\llcorner exp(g, prev(g), 0)) \lrcorner \sim g \uparrow a^{-1} \sim \llcorner exp(g, g, 0)) \lrcorner \uparrow a^{-1}$.

Lemma 9 *Let t,d, and s be ground terms over Σ^{pair}. If $M(t, d, s)$ can be derived from T_C, then $d \in C \cup C^{-1}$ and $\llcorner s \lrcorner \sim \llcorner t \lrcorner \star d$.*

Proof: This lemma is analogical to the previous one, only instead of exponentiation there is multiplication. We need to use functions $incr_i^{mult}$ and $decr_i^{mult}$. □

Lemma 10 *Let r,s, and t be ground terms over Σ^{pair}. If $P(r, s, t)$ can be derived from T_C, then $\llcorner t \lrcorner \sim e(\llcorner r \lrcorner, \llcorner s \lrcorner)$.*

Proof: This equivalence can also be proved by case distinction. We have to look through all possible instances of the predicate P.

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

- $P(r,s,t) = P(mult(x,x_1,\dots,x_m), y, exp(e(x,y),x_1,\dots,x_m))$. We have:
  $\llcorner t \lrcorner = \llcorner exp(e(x,y),x_1,\dots,x_m) \lrcorner$
  $= \llcorner e(x,y) \lrcorner \uparrow c_1^{(t2i(x_1))} \uparrow \dots \uparrow c_m^{(t2i(x_m))}$
  $= e(\llcorner x \lrcorner, \llcorner y \lrcorner) \uparrow c_1^{(t2i(x_1))} \uparrow \dots \uparrow c_m^{(t2i(x_m))}$.
  On the other hand,
  $e(\llcorner r \lrcorner, \llcorner s \lrcorner) = e(\llcorner mult(x,x_1,\dots,x_m) \lrcorner, \llcorner y \lrcorner)$
  $= e(\llcorner x \lrcorner \star c_1^{t2i(x_1)} \star \dots \star c_m^{t2i(x_m)}, \llcorner y \lrcorner)$
  $\sim e(\llcorner x \lrcorner, \llcorner y \lrcorner) \uparrow c_1^{t2i(x_1)} \uparrow \dots \uparrow c_m^{t2i(x_m)}$.

- $P(r,s,t) = P(x, mult(y,y_1,\dots,y_m), exp(e(x,y),y_1,\dots,y_m))$.
  The proof is almost the same, and the only difference is that the arguments of e are switched.

- $P(r,s,t) =$
  $P(mult(x,x_1,\dots,x_m), mult(y,y_1,\dots,y_m), exp(e(x,y),z_1,\dots,z_m))$,
  which requires the predicates $A(x_1,y_1,z_1),\dots,A(x_m,y_m,z_m)$ to be true. In this case, we actually have that $\forall i \; z_i = i2t(t2i(x_i) + t2i(y_i))$, according to the definition of the predicate A.
  $\llcorner t \lrcorner = \llcorner exp(e(x,y),z_1,\dots,z_m)) \lrcorner$
  $= \llcorner e(x,y) \lrcorner \uparrow c_1^{(t2i(i2t(t2i(x_1)+t2i(y_1)))} \uparrow \dots \uparrow c_m^{(t2i(i2t(t2i(x_m)+t2i(y_m)))}$.
  Since $t2i(i2t(t)) = t$, this equals to
  $\llcorner e(x,y) \lrcorner \uparrow c_1^{(t2i(x_1)+t2i(y_1))} \uparrow \dots \uparrow c_m^{(t2i(x_m)+t2i(y_m))}$
  $= e(\llcorner x \lrcorner, \llcorner y \lrcorner) \uparrow c_1^{(t2i(x_1)+t2i(y_1))} \uparrow \dots \uparrow c_m^{(t2i(x_m)+t2i(y_m))}$.
  On the other hand, $e(\llcorner mult(x,x_1,\dots,x_m)) \lrcorner, \llcorner mult(y,y_1,\dots,y_m)) \lrcorner)$
  $= e(\llcorner x \lrcorner \star c_1^{i2t(x_1)} \star \dots \star c_m^{i2t(x_m)}, \llcorner y \lrcorner \star c_1^{i2t(y_1)} \star \dots \star c_m^{i2t(y_m)})$
  $\sim e(\llcorner x \lrcorner, \llcorner y \lrcorner) \uparrow c_1^{i2t(x_1)} \uparrow \dots \uparrow c_m^{i2t(x_m)} \uparrow c_1^{i2t(y_1)} \uparrow \dots \uparrow c_m^{i2t(y_m)}$
  $\sim e(\llcorner x \lrcorner, \llcorner y \lrcorner) \uparrow c_1^{(t2i(x_1)+t2i(y_1))} \uparrow \dots \uparrow c_m^{(t2i(x_m)+t2i(y_m))}$.

- $P(r,s,t) = P(mult(x,x_1,\dots,x_m), mult(y,y_1,\dots,y_m), e(x,y)$. This is just a particular case of the previous rule. Here we have that $\forall i \; z_i = 0$. As in the previous example, we get that
  $\llcorner t \lrcorner = \llcorner x \lrcorner = e(\llcorner x \lrcorner, \llcorner y \lrcorner) \uparrow c_1^{(t2i(x_1)+t2i(y_1))} \uparrow \dots \uparrow c_m^{(t2i(x_m)+t2i(y_m))}$,
  which actually equals to
  $e(\llcorner x \lrcorner, \llcorner y \lrcorner) \uparrow c_1^{(0)} \uparrow \dots \uparrow c_m^{(0)}$
  $\sim e(\llcorner x \lrcorner, \llcorner y \lrcorner)$
  $= e(\llcorner r \lrcorner, \llcorner s \lrcorner)$. □

Example: The most interesting case is when both arguments of pairing function contain multipliers. Let $C = \{a, b\}$. $A(x_1, y_1, z_1), A(x_2, y_2, z_2) \rightarrow P(mult(x_0, x_1, x_2), mult(y_0, y_1, y_2), exp(e(x_0, y_0), z_1, z_2))$ is a rule of T_C. Consider the substitution

$$\sigma = \{g/x_0, 0/x_1, succ(succ(0))/x_2, h/y_0, prev(0)/y_1, prev(0)/y_2\}$$

for some constants g and h. The rule above substitutes $z_1 = prev(0)$, $z_2 = succ(0)$, because only with these values the predicates $A(0, prev(0), prev(0))$ and $A(succ(succ(0)), prev(0), succ(0))$ are true. Then,
$P(mult(g, 0, succ(succ(0))), mult(h, prev(0), prev(0)),$
$exp(e(g, h), prev(0), succ(0)))$
is an instance of T_C. We have:
$\llcorner exp(e(g, h), prev(0), succ(0)) \lrcorner$
$\sim e(g, h) \uparrow a^{-1} \uparrow b$
$\sim e(g \star b^2, h \star a^{-1} \star b^{-1})$
$\sim e(\llcorner mult(g, 0, succ(succ(0))) \lrcorner, \llcorner mult(g, prev(0), prev(0)) \lrcorner)$.

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

Proof: Let $\pi = a_1, \ldots, a_l$ be a (syntactic) derivation for $T_C \vdash a$. The proof proceeds by induction on the length of π. The induction base is $l = 0$, there is nothing to show. For the induction step, we need to show that $\llcorner a_l \lrcorner$ can be derived from $\llcorner \pi_{<l} \lrcorner$, where $\pi_{<l} = a_1, \ldots, a_{l-1}$, and $\pi_{<l}$ is the sequence of atoms obtained from $\pi_{<l}$ by removing all atoms of the form $E(\ldots)$, $M(\ldots)$, and $P(\ldots)$, by replacing all the remaining atoms a_i by $\llcorner a_i \lrcorner$.
By assumption, predicate symbols E,M, and P do not occur in T. It suffices to consider the following cases:

1. If a_l is obtained using integer derivation terms, it must be of the form I(0), I(succ(t)), or I(prev(t)). Therefore, $\llcorner a_l \lrcorner = u$, and we have $T \cup T_E \vdash_E I(u)$ by definition of u.

2. If a_l is obtained using rules:
   $I(x) \rightarrow I(exp(x, 0, \ldots, 0))$,
   $I(exp(x, 0, \ldots, 0)) \rightarrow I(x)$,
   $I(x) \rightarrow I(mult(x, 0, \ldots, 0))$,
   or $I(mult(x, 0, \ldots, 0)) \rightarrow I(x)$,
   it is enough to note that $\llcorner t \lrcorner = \llcorner exp(t, 0, \ldots, 0) \lrcorner = \llcorner mult(t, 0, \ldots, 0) \lrcorner$.

3. If a_l is obtained using $I(c_i), I(y) \rightarrow I(exp(x_0, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_m))$, the atom a_l is of the form $I(exp(s_0, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_m))$ such that

$I(\mathit{exp}(s_0,\dots,s_m))$, $I(c_i)$, and $I(s'_i)$ occur in $\pi_{<l}$.
Set $b = I(\mathit{exp}(s_0,\dots,s_m))$. Then, $\llcorner b \lrcorner = I(\llcorner \mathit{exp}(s_0,\dots,s_m)\lrcorner)$ and $\llcorner I(c_i)\lrcorner = I(c_i)$ are elements of $\llcorner \pi_{<l}\lrcorner$.
Now we need to derive $\llcorner a_l \lrcorner$ from $\llcorner b \lrcorner$ and $I(c_i)$.

- If $t2i(s'_i) > t2i(s_i)$, apply the clause $I(x), I(y) \to I(x \uparrow y)$ from T_E $t2i(s'_i) - t2i(s_i)$ times.
- If $t2i(s'_i) < t2i(s_i)$, apply the clause $I(x) \to I(x^{-1})$ to $I(c_i)$, then apply $I(x), I(y) \to I(x \uparrow y)$ $t2i(s_i) - t2i(s'_i)$ times.
- If $t2i(s'_i) = t2i(s_i)$, then $\llcorner a_l \lrcorner$ is a repetition of $\llcorner b \lrcorner$.

4. If a_l is obtained using $I(c_i), I(y) \to I(\mathit{mult}(x_0,\dots,x_{i-1},y,x_{i+1},\dots,x_m))$, the atom a_l is of the form $I(\mathit{mult}(s_0,\dots,s_{i-1},s'_i,s_{i+1},\dots,s_m))$ such that $I(\mathit{mult}(s_0,\dots,s_m))$, $I(c_i)$, and $I(s'_i)$ occur in $\pi_{<l}$. The derivation in this case is analogical to exponentiation:

   - If $t2i(s'_i) > t2i(s_i)$, apply the clause $I(x), I(y) \to I(x \star y)$ $t2i(s'_i) - t2i(s_i)$ times.
   - If $t2i(s'_i) < t2i(s_i)$, apply the clause $I(x) \to I(x^{-1})$ to $I(c_i)$, then apply $I(x), I(y) \to I(x \star y)$ $t2i(s_i) - t2i(s'_i)$ times.
   - If $t2i(s'_i) = t2i(s_i)$, then $\llcorner a_l \lrcorner$ is a repetition of $\llcorner b \lrcorner$.

5. If a_l is obtained using the rule:
   $A(x_1,y_1,z_1),\dots,A(x_m,y_m,z_m)$,
   $I(\mathit{mult}(x,x_1,\dots,x_m)),\dots,I(\mathit{mult}(y,y_1,\dots,y_m)) \to$
   $I(\mathit{exp}(e(x,y),z_1,\dots,z_m))$,
   then the atom a_l is of the form $\mathit{exp}(e(u_0,v_0),s_1,\dots,s_m)$, such that $I(\mathit{mult}(u_0,u_1,\dots,u_m))$, $I(\mathit{mult}(v_0,v_1,\dots,v_m))$ occur in $\pi_{<l}$, and the terms $A(u_1,v_1,s_1),\dots,A(u_m,v_m,s_m)$ are true. By definition of the predicate A, they are true iff $s_i = i2t(t2i(u_1) + t2i(v_i))$ for each i.
   Set $b_1 = I(\mathit{mult}(u_0,u_1,\dots,u_m))$, and $b_2 = I(\mathit{mult}(v_0,v_1,\dots,v_m))$.
   $\llcorner b_1 \lrcorner = I(\llcorner \mathit{mult}(u_0,u_1,\dots,u_m)\lrcorner)$ and
   $\llcorner b_2 \lrcorner = I(\llcorner \mathit{mult}(v_0,v_1,\dots,v_m)\lrcorner)$
   are elements of $\llcorner \pi_{<l}\lrcorner$. Then, $\llcorner a_l \lrcorner$ can be derived from $\llcorner b_1 \lrcorner$ and $\llcorner b_2 \lrcorner$ by applying the rule $I(x), I(y) \to I(e(x,y))$.
   We get a term $e(\dots,\dots)$, without any exponents, but using equations $e(x, y \star z) = e(x,y) \uparrow z$ and $e(x,y) = e(y,x)$ we get that the result is actually equivalent to $\llcorner a_l \lrcorner$. Each exponent c_i is taken out from $\llcorner b_1 \lrcorner$ u_i times, and from $\llcorner b_2 \lrcorner$ v_i times. As the result, we get $\llcorner \mathit{exp}(e(u_0,v_0), i2t(t2i(u_1)+t2i(v_1)),\dots,i2t(t2i(u_m)+t2i(v_m)))\lrcorner =$
   $\llcorner \mathit{exp}(e(u_0,v_0),s_1,\dots,s_m))\lrcorner = \llcorner a_l \lrcorner$.

6. Suppose that a_l is obtained using the rule:
$M(\ulcorner\theta(s'_1)\urcorner, b_1, x_1), \ldots, M(\ulcorner\theta(s'_j)\urcorner, b_j, x_j)$,
$E(\ulcorner\theta(t'_1)\urcorner, d_1, y_1), \ldots, E(\ulcorner\theta(t'_k)\urcorner, d_k, y_k)$,
$P(\ulcorner\theta(u'_1)\urcorner, \ulcorner\theta(v'_1)\urcorner, z_1), \ldots, P(\ulcorner\theta(u'_l)\urcorner, \ulcorner\theta(v'_1)\urcorner, z_l)$,
$\ulcorner\theta(r_1)\urcorner, \ldots, \ulcorner\theta(r_n)\urcorner \to \ulcorner\theta(r_0)\urcorner$.

   Assume that this clause was instantiated with a substitution σ: it means that $a_l = \sigma(\ulcorner r_0 \urcorner)$. Furthermore, all the $\sigma(\ulcorner\theta(r_i)\urcorner)$ for all $i \in \{1, \ldots, n\}$, and all the $E(\sigma(\ulcorner\theta(t'_i)\urcorner), d_i, \sigma(y_i))$, $M(\sigma(\ulcorner\theta(s'_i)\urcorner), b_i, \sigma(x_i))$, and $P(\sigma(\ulcorner\theta(u'_i)\urcorner), \sigma(\ulcorner\theta(v'_i)\urcorner), \sigma(z_i))$ for all $i \in \{1, \ldots, k\}$, are in $\pi_{<l}$. Therefore, $\llcorner\sigma(\ulcorner\theta(r_i)\urcorner)\lrcorner$ for all $i \in \{1, \ldots, n\}$ are in $\llcorner\pi_{<l}\lrcorner$.
   By Lemma 8, we have $\llcorner\sigma(y_i)\lrcorner \sim \llcorner\sigma(\ulcorner\theta(t'_i)\urcorner)\lrcorner \uparrow d_i$.
   By Lemma 9, we have $\llcorner\sigma(x_i)\lrcorner \sim \llcorner\sigma(\ulcorner\theta(s'_i)\urcorner)\lrcorner \star b_i$.
   By Lemma 10, we have $\llcorner\sigma(z_i)\lrcorner \sim e(\llcorner\sigma(\ulcorner\theta(u'_i)\urcorner)\lrcorner, \llcorner\sigma(\ulcorner\theta(v'_i)\urcorner)\lrcorner)$.
   Let $\sigma^*(x) = \llcorner\sigma(x)\lrcorner$. For each subterm t of $r_0, \ldots, r_n$ such that t is not of the form w^{-1}, we show by induction on the size of t, that $\sigma^*(t) \sim \llcorner\sigma(\ulcorner\theta(t)\urcorner)\lrcorner$:

   (a) If $t = x$ is a variable: $\ulcorner\theta(x)\urcorner = x$, and thus $\sigma^*(x) = \llcorner\sigma(\ulcorner\theta(x)\urcorner)\lrcorner$, by definition of σ^*.

   (b) If $t = f(t_1, \ldots, t_n)$ for $f \notin \{\uparrow, \star\}$: the claim easily follows by induction.

   (c) If $t = t' \uparrow d$ and t is ground: $\llcorner\sigma(\ulcorner\theta(t)\urcorner)\lrcorner = \llcorner\ulcorner t \urcorner\lrcorner$, and $\sigma^*(t) = t$. We know that $\llcorner\ulcorner t \urcorner\lrcorner = t$ by Lemma 7.

   (d) If $t = t' \star d$ and t is ground: $\llcorner\sigma(\ulcorner\theta(t)\urcorner)\lrcorner = \llcorner\ulcorner t \urcorner\lrcorner$, and $\sigma^*(t) = t$. We know that $\llcorner\ulcorner t \urcorner\lrcorner = t$ by Lemma 7.

   (e) If $t = t_i = t'_i \uparrow d_i$: we have $\sigma^*(t_i) = \sigma^*(t'_i) \uparrow d_i \sim \llcorner\sigma(\ulcorner\theta(t'_i)\urcorner)\lrcorner \uparrow d_i$, by the induction hypothesis. We have $\llcorner\sigma(y_i)\lrcorner \sim \llcorner\sigma(\ulcorner\theta(t'_i)\urcorner)\lrcorner \uparrow d_i$. Therefore, $\sigma^*(t_i) \sim \llcorner\sigma(y_i)\lrcorner = \llcorner(\ulcorner\theta(t_i)\urcorner)\lrcorner$.

   (f) If $t = s_i = s'_i \star b_i$: we have $\sigma^*(s_i) = \sigma^*(s'_i) \star b_i \sim \llcorner\sigma(\ulcorner\theta(s'_i)\urcorner)\lrcorner \uparrow b_i$, by the induction hypothesis. We have $\llcorner\sigma(x_i)\lrcorner \sim \llcorner\sigma(\ulcorner\theta(s'_i)\urcorner)\lrcorner \star b_i$. Therefore, $\sigma^*(s_i) \sim \llcorner\sigma(x_i)\lrcorner = \llcorner(\ulcorner\theta(s_i)\urcorner)\lrcorner$.

   (g) If $t = w_i = e(u'_i, v'_i)$: we have $\sigma^*(w_i) = e(\sigma^*(u'_i), \sigma^*(v'_i)) \sim e(\llcorner\sigma(\ulcorner\theta(u'_i)\urcorner)\lrcorner, \llcorner\sigma(\ulcorner\theta(v'_i)\urcorner)\lrcorner)$, by the induction hypothesis. We have $\llcorner\sigma(z_i)\lrcorner \sim e(\llcorner\sigma(\ulcorner\theta(u'_i)\urcorner)\lrcorner, \llcorner\sigma(\ulcorner\theta(v'_i)\urcorner)\lrcorner)$. Therefore, $\sigma^*(w_i) \sim \llcorner\sigma(z_i)\lrcorner = \llcorner(\ulcorner\theta(w_i)\urcorner)\lrcorner$.

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

we can apply the clause $r_1, \dots, r_n \rightarrow r_0$ with σ^* to obtain $\sigma^*(r_0) \sim \llcorner\sigma(\ulcorner\theta(r_0)\urcorner)\lrcorner = \llcorner a_l \lrcorner$. $\square$

Proof of Theorem 2: Now, suppose that $T_C \vdash \ulcorner b \urcorner$. By assumption, $b = p(t)$, where p occurs in T. Lemma 11 implies that $T \cup T_E \vdash_E \llcorner\ulcorner b \urcorner\lrcorner$. By Lemma 7, $\llcorner\ulcorner b \urcorner\lrcorner \sim b$, and therefore $T \cup T_E \vdash_E b$. $\square$

As the result, we get that any cryptographic protocol that uses bilinear pairings with products in exponents can be encoded by some pre-written protocol transformer and derived to a form that is suitable for ProVerif. We have proven that if ProVerif proves that the encoded protocol is secure, it implies that the actual protocol is also secure. If ProVerif finds a flaw in it, it shows which rules the intruder applied with the encoded protocol. We have analogical rules in the theory T_E, and these are the rules that would be applied by the intruder in the reality.

4 Implementation

4.1 Implementation of the Protocols Fully Supported by the Theory

The Diffie-Hellman protocol transformer that was used in the [6] was upgraded. There are now two versions of the protocol transformer for protocols with using types and without using types. The protocol is written first as an ordinary Prolog program, and afterwards it is translated to a file that can be tested by ProVerif. The version that supports types cannot yet check if all the functions that are present in the protocol are actually predefined, since the syntax is slightly more complex.

Several protocols that are using bilinear mappings have been tested in ProVerif. All of them are key-agreement protocols. More complex protocols have been first written in pi-calculus, and afterwards converted to Horn clauses and processed by a program that converts theory T to theory T_C.

Let G_1 be a group generated by P. Let the generator P and the size of the group be known to everyone. Let a, b, and c be the nonces generated by parties A, B, and C.

TEST 1 *Joux's protocol [5]*

- $A \to B, C : aP$
- $B \to A, C : bP$
- $C \to A, B : cP$

The three parties may then calculate the key by following:

- A gets $e(bP, cP)^a$.
- B gets $e(aP, cP)^b$.
- C gets $e(aP, bP)^c$.

According to bilinearity, the new key is $e(P, P)^{abc}$.

It was tested for both authenticated and non-authenticated channels. It is vulnerable if the channels are non-authenticated because the active adversary may substitute coming values aP, bP, and cP with whatever he wants. It is secure for authenticated channels.

TEST 2 *Shim's protocol variation [4]*

This protocol requires a little bit more variables. It takes into account secret and public keys of the parties. Let x, y, and z be the secret keys of A, B, and C. Let xP, yP, and zP be the corresponding public keys. The public keys are found in the certificates.

- $A \rightarrow B, C : aP, a(xP), CertA$
- $A \rightarrow B, C : bP, b(yP), CertB$
- $A \rightarrow B, C : cP, c(zP), CertC$

- A checks: $e(bP, yP) == e(b(yP), P), e(cP, zP) == e(c(zP), P)$.
- B checks: $e(aP, xP) == e(a(xP), P), e(cP, zP) == e(c(zP), P)$.
- C checks: $e(bP, yP) == e(b(yP), P), e(aP, xP) == e(a(xP), P)$.

- A gets $e(b(yP), c(zP))^{ax}$.
- B gets $e(a(xP), c(zP))^{by}$.
- C gets $e(b(yP), a(xP))^{cz}$.

According to bilinearity, the new key is $e(P, P)^{abcxyz}$.

The most tricky place in this protocol is the comparison. We cannot simply use equation, we need equivalnce instead. The current version of protocol transformer does not support comparison of two terms *mod E* inside the protocol. We had to add the comparision through a newly defined variable. For example, in order to compare $e(x, y)$ and $e(w, z)$, we write $P(x, y, v) \,\&\, P(w, z, v)$. It is true iff $e(x, y) \sim e(w, z)$, according to the definition of P. If we do not use the comparison, then ProVerif claims the protocol to be insecure. It is secure if we do compare the appropriate values.

TEST 3 *TAK 1 [1]*

This protocol again uses public and private keys of the parties. Additionally, we will need to define a hash function H.
This protocol was first written in pi-calculus and afterwards converted to Horn clauses and transformed. In this case, we did not need to add anything by hand. It does not use comparison.

- $A \rightarrow B, C : aP, CertA$
- $A \rightarrow B, C : bP, CertB$

- $A \to B, C : cP, CertC$

- A gets $H((e(bP, cP)^a, e(yP, zP)^x))$.
- B gets $H((e(aP, cP)^b, e(xP, zP)^y))$.
- C gets $H((e(aP, bP)^c, e(xP, yP)^z))$.

The new key is $H((e(P, P)^{abc}, e(P, P)^{xyz}))$.

There were no problems in this protocol, and it turned out to be secure.

TEST 4 *A Six Pass Pairing Based AKC Protocol [1]*

This protocol also makes use of the public and private keys, but it does not include these keys in the newly generated symmetric key.

The structure of the given protocol is a little bit more complex than the previous protocols. It was really necessary to write it first in pi-calculus to avoid mistakes.

- $A \to B : aP, CertA$
- $B \to C : aP, CertA, bP, CertB$
- $C \to A : bP, CertB, cP, CertC, enc(K_{abc}, sign(K_c, (A, B, aP, bP, cP))$
- $A \to B : cP, CertC, enc(K_{abc}, sign(K_c, (A, B, aP, bP, cP))),$
  $enc(K_{abc}, sign(K_a, (B, C, aP, bP, cP)))$
- $B \to C : enc(K_{abc}, sign(K_a, (B, C, aP, bP, cP))),$
  $enc(K_{abc}, sign(K_b, (A, C, aP, bP, cP)))$
- $B \to A : enc(K_{abc}, sign(K_b, (A, C, aP, bP, cP)))$

The new key will be $e(P, P)^{abc}$.

ProVerif needed some time to analyze the protocol because each party sends something into the net several times. In the end, it turned out to be secure.

TEST 5 *TAK 2' variation protocol [1]*

This protocol turned out to be a big problem for our solution. It uses the pairing function three times in the key. Although the analysis process is convergent, it takes too much time for ProVerif to analyze it. We had to test its security for each party separately. This should not affect the result since this is a one-round protocol.

- $A \to B, C : aP, CertA$
- $A \to B, C : bP, CertB$
- $A \to B, C : cP, CertC$

- A gets $H(e(bP, zP)^a, e(cP, yP)^a, e(bP, cP)^x)$.
- B gets $H(e(aP, zP)^b, e(aP, cP)^y, e(cP, xP)^b)$.
- C gets $H(e(aP, bP)^z, e(aP, yP)^c, e(bP, xP)^c)$.

The new key is $H(e(P, P)^{abz}, e(P, P)^{acy}, e(P, P)^{bcx})$. ProVerif has discovered a vulnerability in this protocol, that was described in [8]. Although the parties are using certificates, they may only check if xP, yP, and zP are the proper values. If nothing is done to the authentication of the values aP, bP, and cP, the intruder may substitute them with something else. In $TAK1$, the intruder was unable to construct the key since it contained the element $e(yP, zP)^x$, where both arguments are some public keys that require certification.

TEST 6 *A simple ID-based pairing protocol*

This protocol allows a quick and simple generation of symmetric keys between two parties. Here we use definitions diferent from the protocols above.

Let $h(A)$ be the public key of party A, and $h(B)$ the public key of the party B. Moreover, the values $h(B)$ and $h(A)$ are the points on some elliptic curve. Let s be the master secret that is known only to a trusted third party, the CA server. A party may apply for a secret key, and CA returns $h(A) \star s$ without telling the value s. The distribution of secret keys is not a part of the protocol.

If two parties, A and B, decide to communicate, they do not need to exchange anything in the net. Each party may calculate the corresponding key:

- A generates a key $e(h(A) \star s, h(B))$.
- B generates a key $e(h(B) \star s, h(A))$.

The attacker cannot do anything since the parties generate the keys themselves, and the identities are included in the key. This protocol was proved to be secure.

4.2 Implementation of the Protocols Partially Supported by the Theory

Unfortunately, for many protocols it is not sufficient to use only products in exponents. There are some protocols that are using addition of the exponents. We need to define additional equations in our theory. Let $+$ denote addition in G_1, and $\cdot$ multiplication in G_T.

We need to introduce the identity elements in the groups. Let 0 be the identity element in G_1, and 1 the identity element in G_T.

1. $x + (y + z) = (x + y) + z$ for all $x, y, z \in G_1$;
2. $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ for all $x, y, z \in G_T$;
3. $e(x, y + z) = e(x, y) \cdot e(x, z)$ for all $x, y, z \in G_1$;
4. $x + x^{-1} = 0$ for each $x \in G_1$;
5. $x \cdot x^{-1} = 1$ for each $x \in G_T$;
6. $e(x, 0) = 1$ for each $x \in G_1$.

The problem of the first two rules is that we need to define associativity, and ProVerif is not good at it. We cannot also define a finite set of rules like in the case of exponentiation, since the elements of G_1 and G_T may be non-ground variables.

One thing that we may do is to limit the number of terms in the sum (product). We can implement it in the case when a protocol does not use more terms in the additions.

We define additional rules that may be used by the intruder (this theory is not yet proved). It would be better to use a special notation for the identity elements and write $I(0)$ and $x + x^{-1} = 0$ instead of $I(x + x^{-1})$, but these kinds of equations are not suitable for ProVerif.

1. $I(x), I(y) \rightarrow I(x + y)$;
2. $I(x), I(y) \rightarrow I(x \cdot y)$;
3. $I(x), I(x + y) \rightarrow I(y)$;
4. $I(x), I(x \cdot y) \rightarrow I(y)$;
5. $I(x + x^{-1})$;
6. $I(y \cdot y^{-1})$;

7. $I(e(x, y + y^{-1}))$.

When describing rules for encoded terms, we need the rule $I(x) \to I(x^{-1})$ that we have eliminated in the previous definition of theory T_C. We introduce it back since now we use inverses not only of exponent-ground integers.

First, we need a function for inverses in groups. Let these functions be inv^{G_1} for G_1 elements, and inv^{G_T} for G_T elements. Then we define the functions for the group operations(as we did it for exponentiation and multiplication). Let these functions be add^{G_1} for addition in G_1 and $mult^{G_T}$ for multiplication in G_T. Without associativity, we may implement only functions of arity 2. The additional rules in theory T_C would then be:

1. $I(x), I(y) \to I(add^{G_1}(x, y))$;
2. $I(x), I(y) \to I(mult^{G_T}(x, y))$;
3. $I(x), I(add^{G_1}(x, y)) \to I(y)$;
4. $I(x), I(mult^{G_T}(x, y)) \to I(y)$;
5. $I(x) \to I(inv^{G_1}(x))$;
6. $I(x) \to I(inv^{G_T}(x))$;
7. $I(inv^{G_1}(x)) \to I(x)$;
8. $I(inv^{G_T}(x)) \to I(x)$.

The facts below hold for any $x \in G_1$, since the terms are actually equivalent to some identity elements.

1. $I(add^{G_1}(x, inv^{G_1}(x)))$;
2. $I(mult^{G_T}(x, inv^{G_T}(x)))$;
3. $I(e(x, add^{G_1}(y, inv^{G_1}(y))))$.

Additionally, we have to add some rules for associativity of group operations. The equation like

$$add^{G_1}(x, add^{G_1}(y, z)) = add^{G_1}(add^{G_1}(x, y), z)$$

does not work in ProVerif. If we have limited number of exponents, then it will be easier. If the protocol adds only two terms, we may define commutativity instead of associativity:

$$add^{G_1}(x, y) = add^{G_1}(y, x) \ .$$

These rules do not fully describe the actual theory for addition in exponents, but they are better than nothing, and this theory will be developed in future research.

There is no automatic protocol transformer for this theory, so that the tested protocols have been written directly in ProVerif Horn clauses, without being preprocessed. The only thing that was generated automatically is the set of T_C rules for the intruder using the same number of grounded exponents.

TEST 7 *Smart's ID-based AK Protocol [9]*

This is a protocol that uses two terms in the addition, and therefore can be tested with our unproved theory. The main idea of this protocol is based on the identifiers of parties.

The pairing is performed in a group $G_1 = \langle P \rangle$. Let ID_A be the identifier string of party A, and ID_B the identifier of party B. Let some hash function $H : ID \to G_1$ be applied to the values

$$Q_a = H(ID_A)\ ,$$

$$Q_b = H(ID_B)\ .$$

There is some key distribution centre that generates a public/private key pair. It has generated some master secret key s. Each party gets its private secret key, such that nobody except the key distribution centre knows the value of s: $S_a = s * Q_a$, and $S_b = s * Q_b$. Everybody knows the values P and $P_s = s * P$.

When starting to communicate, the party A generates a random number a, and party B generates a random number b. Then, A and B compute $T_a = aP$ and $T_b = bP$, and send these values into the net. Both parties may then calculate a new shared key:

- A gets $e(S_a, T_b) \cdot e(aQ_b, P_s) = e(bQ_a, sP) \cdot e(aQ_b, sP) = e(aQ_b + bQ_a, sP)$.
- B gets $e(S_b, T_a) \cdot e(bQ_a, P_s) = e(aQ_b, sP) \cdot e(bQ_a, sP) = e(aQ_b + bQ_a, sP)$.

The current realization of this protocol in ProVerif that uses only two terms in addition did not find any vulnerabilities.

TEST 8 *A More Efficient Identity Based Authenticated Key Agreement Protocol [3]*

This protocol is very similar to Smart's ID-based AK Protocol, but its computation is more efficient. Instead of exchanging aP and bP, the parties exchange the values $W_a = aQ_a$ and $W_b = bQ_b$. The parties may then compute the shared key:

- A gets $e(S_a, W_b + aQ_b) = e(Q_a, Q_b)^{s(a+b)}$.
- B gets $e(S_b, W_a + bQ_a) = e(Q_a, Q_b)^{s(a+b)}$.

This protocol has also been tested, and ProVerif found the vulnerability in one particular case when $W_b = aQ_b^{-1}(W_b^{-1} = aQ_b)$ or $W_a = bQ_a^{-1}(W_a^{-1} = bQ_a)$. This, however, may happen with a negligible probability, since the attacker cannot compute the inverse of either aQ_b or bQ_a. No more problems were found yet.

4.3 Efficiency of the Analyzer

It has been tested, how much time it takes for ProVerif to analyze the following protocols. The tests were performed at 2.21 GHz AMD Athlon 64X2 Dual Core Processor 4400+ with 2.00 GB of RAM. The protocols were translated by the version of protocol transformer that does not use types. Afterwards, they were tested by ProVerif 1.84. The time would be longer with ProVerif 1.85.

It was also necessary to translate the theory T into theory T_C, but this time is too small if compared to the time that was spent by ProVerif on protocol analysis. This time is not presented in this table since it is not so important.

Protocol name	**Nr.of tests**	**Average time (sec)**
A simple ID-based pairing protocol	1000	0.0149
A More Efficient Identity Based Authenticated Key Agreement Protocol [3]	1000	0.0310
Smart's ID-based AK Protocol [9]	1000	0.0508
Joux's protocol [5]	100	0.273
TAK 1 [1]	10	254
Shim's protocol variation [4]	10	836
A Six Pass Pairing Based AKC Protocol [1]	10	1330
TAK 2' variation protocol [1]	1	> 43200

5 Conclusions and Future Work

As the result of this research, it was shown that it is possible to implement bilinear mappings in protocol analyzers that are based on predicate logic. Several cryptographic protocols were proven to be secure in the range of C-exponent-ground theory for bilinear mappings with products in exponents.

The analysis of cryptographic protocols in ProVerif using given theory is very slow. It takes too much time to check, for example, the protocol that uses three pairings in one message, or the protocol that uses ten different grounded exponents. This is primarily caused by complexity of addition of integers. Also, there are too many rules that allow to get elements from G_T: the intruder may derive them not only by multiplication and division in G_T, but also by pairing, that in turn uses several different rules. This solution is inefficient and allows to test only the simplest protocols, but it is still useful since too complex protocols are computationally inefficient themselves, and for the simpler protocols the current version is sufficient.

If a protocol is being analyzed using type constraints (multiplication allowed only in G_1, exponentiation only in G_T), it somehow works even slower. It could mean that ProVerif does not apply group operations to wrong elements even if the types are not defined.

This work definitely requires future research. It is necessary to compose an appropriate theory for addition of exponents and formally prove it. Something could be done with associativity: although these kinds of equations are non-convergent in ProVerif, it is probably possible to do something with a finite number of multipliers (like it was in case of grounded exponentiation). It is necessary to write an automatic transformation for protocols using addition in exponents. It would be nice to write a single transformer that would work at once with and without types, and could find all the possible errors in the description of the protocol (the current version of protocol transformer that uses types does not have special syntax error control).

Bilineaarsed kujutused formaalses krüptograafias

Bakalaureusetöö(6 EAP)
Alisa Pankova
Lühikokkuvõte

Krüptograafiliste protokollide turvalisuse testimiseks on loodud erinevad analüsaatorid. Osa neist põhineb predikaatloogika valemitel. Formaalses mudelis pole aga mugav realiseerida aritmeetilisi funktsioone. On kerge arvutada g^a, kui on teada nii g kui a väärtused, kuid protokollides on muutujad üldjuhul väärtustamata. Algebraliste struktuuride omadusi on vaja kirjeldada loogika valemite abil. Mõnede sellist liiki probleemidega on juba tegeldud. Näiteks on realiseeritud Diffie-Hellmani astendamine Horni valemitel põhineva analüsaatoriga ProVerif [2] [6]. Kahjuks see töötab vaid erinevate astendajate lõpliku arvu korral.

Peale astendamist pakuvad aga krüptograafia valdkonnale huvi ka muud algebralised struktuurid, nende hulgas ka bilineaarsed kujutused [7]. Antud uurimistöö eesmärk oli realiseerida bilineaarsete kujutuste arvutamist analüsaatoriga ProVerif ning analüüsida moodustatud protokolliteisendaja abil mõningaid bilineaarseid kujutusi kasutavaid protokolle.

Artiklis [6] toodud teooria on selles töös laiendatud bilineaarsetele kujutustele. On kirjeldatud teooria, mis sisaldab reegleid bilineaarsuse omaduse jaoks, et analüsaator saaks aru, et näiteks $e(aR, bS) = e(abR, S)$. Lahenduse põhiidee on kanda üle kõik kordajad rühmast G_1 rühma G_T . Siis saame näiteks, et $e(aR, bS) = e(R, S)^{ab}$ ja $e(abR, S) = e(R, S)^{ab}$, nende avaldiste paremad pooled on süntaktiliselt võrdsed.

Bilineaarsete kujutuste realiseerimine predikaatloogika abil ei ole aga efektiivne. Kui erinevate astendajate arv ligineb kümnele, siis on vaja päris kaua oodata, et analüsaator oma töö lõpetaks. Analüüsi aeglus on põhjendatud suure hulga variantidega. Näiteks $e(R, S)^a$ on tuletatav mitte ainult valemitest $e(aR, S)$ ja $e(R, aS)$, vaid lõpmata paljudest teistest valemitest: $e(abR, ab^{-1}S)$, $e(aaa^{-1}R, S)$ jne. Sama probleem on ka lihtsalt Diffie-Hellmani astendamisega, kuid bilineaarsete kujutuste korral lisab igale variandile keerukust ka astendajate liitmine.

Üldiselt on aga saavutatud eesmärk, et bilineaarsete kujutuste omaduste realiseerimine on võimalik ning mõned protokollid on ka testitud. On kirjutatud automaatne protokolliteisendaja nii tüüpidega kui ilma tüüpideta variantide jaoks. Selle abil on testitud mõnede bilineaarseid funktsioone kasutatavate protokollide turvalisus.

References

[1] Sattam S. Al-Riyami and Kenneth G. Paterson. Tripartite Authenticated Key Agreement Protocols from Pairings. Cryptology ePrint Archive, Report 2002/035, 2002. `http://eprint.iacr.org/`.

[2] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *CSFW*, pages 82–96. IEEE Computer Society, 2001.

[3] Liqun Chen and Caroline Kudla. Identity Based Authenticated Key Agreement Protocols from Pairings. Cryptology ePrint Archive, Report 2002/184, 2002. `http://eprint.iacr.org/`.

[4] Zhaohui Cheng, Luminita Vasiu, and Richard Comley. Pairing-Based One-Round Tripartite Key Agreement Protocols. Cryptology ePrint Archive, Report 2004/079, 2004. `http://eprint.iacr.org/`.

[5] Antoine Joux. A One Round Protocol for Tripartite Diffie-Hellman. *J. Cryptology*, 17(4):263–276, 2004.

[6] Ralf Küsters and Tomasz Truderung. Using ProVerif to Analyze Protocols with Diffie-Hellman Exponentiation. In *CSF*, pages 157–171. IEEE Computer Society, 2009.

[7] Alfred Menezes. An introduction to pairing-based cryptography. In Ignacio Luengo, editor, *Recent Trends in Cryptography*, volume 477 of *Contemporary Mathematics*, pages 47–65. AMS, 2009.

[8] Kyungah Shim. Cryptanalysis of Al-Riyami-Paterson's Authenticated Three Party Key Agreement Protocols. Cryptology ePrint Archive, Report 2003/122, 2003. `http://eprint.iacr.org/`.

[9] Nigel P. Smart. An Identity Based Authenticated Key Agreement Protocol Based on the Weil Pairing. Cryptology ePrint Archive, Report 2001/111, 2001. `http://eprint.iacr.org/`.

Appendix — CD containing the sources of the transformer

The attached CD contains the modified transformer and the specifications of the tested protocols. It also contains a README-file explaining how to build and run the transformer and the analyzer.

The batch file "timer.bat" was used to run efficiency tests on Windows platform. The file "timer.txt" contains the results of the test.