Unification in Blind Signatures

Serdar Erbatur* [1], Christopher Lynch † [2], and Paliath Narendran‡ [1]

[1] University at Albany–SUNY {se,dran}@cs.albany.edu
[2] Clarkson University clynch@clarkson.edu

Abstract

Blind signatures are signature schemes that keep the content confidential and have applications in modern cryptography for electronic voting and digital cash schemes. We study three unification problems based on an equational theory for blind signatures. This theory consists of two axioms, namely

$$\begin{aligned} U(S(B(m,x)),x) &= S(m) && (\mathcal{E}_1) \\ m * B(n,r) &= B(m*n,r) && (\mathcal{E}_2) \end{aligned}$$

derived from its implementation with RSA. First, the unification problem modulo $\mathcal{E}_1$ is shown to be NP-complete and of type finitary. An algorithm based on deduction rules is given. Second, unification in $\mathcal{E}_2$ is shown to be decidable and of type unitary. Likewise, we give an algorithm which returns a unique unifier if there exists one and provide necessary failure rule mechanisms to detect function clash, occur-check and infinite splitting. Finally, the combination of unification problems $\mathcal{E}_1$ and $\mathcal{E}_2$ turns out to be decidable. The result follows from techniques of equational term rewriting systems and unification in the subtheories $\mathcal{E}_1$ and $\mathcal{E}_2$. Consequently, these results are useful for symbolic analysis of protocols deploying blind signatures.

1 Introduction

In formal cryptographic protocol analysis, messages are represented as terms, where the functions in the terms represent actions that can be performed on messages, such as encrypting a message with some key, hashing a message, or calculating the exclusive OR of two messages.

A protocol is described formally by the actions of a principal, who will receive a message, and then send out another message based on the message received. An attack can be represented by the intruder learning some secret. A tool for cryptographic protocol analysis can then work its way back from the goal to initial facts, to see if a sequence of actions which leads to the intruder learning the secret message is possible. At each stage of this search, unification must be performed between terms representing messages sent and terms representing messages that are expected to be received. These terms may contain variables representing unknowns.

Traditionally, cryptographic protocol analysis tools work in the free algebra, which gives no meaning to the function symbols, and terms can only be equal if they are syntactically the same. However, tools such as the Maude NPA [7] can give a deeper analysis. Equational properties of terms can be given, which take into account the meaning of a function symbol. Then unification can be performed modulo the equational theory.

For example, consider the case of blind signatures. There is a blinding function B, which blinds a message m with a given key x. We can represent this by the term $B(m,x)$. There is an unblinding function U which performs the inverse of unblinding for some key. There is also a signing function. Blind signatures have the

*Partially supported by the NSF grants CNS-0831209 and CNS-0905286
†Partially supported by the NSF grants CNS-0831305 and CNS-0905378
‡Partially supported by the NSF grants CNS-0831209 and CNS-0905286

property that if a message m is blinded with some key x, then signed, and then unblinded with x, the signed message will emerge. These are called blind signatures, because the signer could not tell what was being signed. Blind signatures are used in electronic voting and digital cash [5]. The properties we have just described can be represented by the following equation:

$$U(S(B(m,x)),x) = S(m)$$

Blind RSA signatures [4] are created by multiplying the message m by a random number r raised to a public key e. Therefore, multiplying an RSA-blinded message by another number (message) is equivalent to multiplying the two messages and then blinding them.

The two properties mentioned above are represented by the following axioms:

$$U(S(B(m,x)),x) = S(m) \quad (1)$$

$$m * B(n,r) = B(m * n, r) \quad (2)$$

We denote axiom (1) as equational theory $\mathcal{E}_1$ and axiom (2) as $\mathcal{E}_2$.

Performing unification modulo $\mathcal{E}_1$ and $\mathcal{E}_2$ will allow a cryptographic analysis tool to give a deeper analysis of a cryptographic protocol which uses blinding. Therefore, in this paper we give unification algorithms for $\mathcal{E}_1$, $\mathcal{E}_2$ and the theory consisting of both of them.

The algorithms given are based on inference rules originally given in [14], which gave a unification algorithm for one-sided distributivity. We give an algorithm to generate a complete set of unifiers for each of these theories. The algorithm for $\mathcal{E}_1$ runs in nondeterministic polynomial time and gives a finite complete set of unifiers. The algorithm for $\mathcal{E}_2$ gives a single most general unifier, and the algorithm for the combination of the two also gives a finite complete set of unifiers.

We describe the algorithm for $\mathcal{E}_1$ in Section 3, for $\mathcal{E}_2$ in Section 4, and for the combination in Section 5.

2 Preliminaries

We introduce some basic definitions here. The reader is referred to the survey [2] for more details.

Let $\mathcal{S} = \{s_1 =_E^? t_1, \ldots, s_m =_E^? t_m\}$ be an E-unification problem. An *E-unifier* for $\mathcal{S}$ is a substitution σ such that $\sigma(s_i) =_E \sigma(t_i)$ for all $1 \leq i \leq m$. That is, *equality modulo E* , $=_E$, in $\mathcal{S}$ is satisfied if we apply σ to every equation. The set of all E-unifiers of $\mathcal{S}$ is denoted by $U_E(\mathcal{S})$. It is said that σ is *more general modulo E* than θ on a set of variables V, denoted as $\sigma \leq_E \theta$, if and only if there is a substitution τ such that $\sigma\tau(x) =_E \theta(x)$ for all $x \in V$. A *complete set of E-unifiers* of $\mathcal{S}$ is a set Σ of substitutions such that every $\theta \in \Sigma$ is an E-unifier and for every E-unifier θ, there is a substitution $\sigma \in \Sigma$ where $\sigma \leq_E \theta$ holds. A complete set of E-unifiers Σ of a unification problem $\mathcal{S}$ is said to be *minimal* if and only if for any two E-unifiers σ and θ in Σ, $\sigma \leq_E \theta$ implies that $\sigma =_E \theta$.

An E-unification problem $\mathcal{S}$ is of type *unitary*, if the minimal complete set of E-unifiers of $\mathcal{S}$ has size one. $\mathcal{S}$ is *finitary (infinitary)* if the minimal complete set of E-unifiers of it is finite (infinite). We note that the minimal set of unifiers might be empty even when the problem is unifiable. We say $\mathcal{S}$ is of type *zero* in that case. An equational theory E is *unitary* if the maximal type of an E-unification problem is unitary. Similarly, E is *finitary* if E-unification problems have at most type finitary. If there exists a problem of type infinitary on E and no problem of type nullary, then E is *infinitary*. E has type *zero* (or E is *nullary*) if it has a problem of type zero.

A set of equations (i.e., a unification problem) is said to be in *dag-solved form* (or *d-solved form*) if and only if they can be arranged as a list

$$x_1 =^? t_1, \ \ldots, \ x_n =^? t_n$$

where (a) each left-hand side x_i is a distinct variable, and (b) $\forall\, 1 \leq i \leq j \leq n$: x_i does not occur in t_j ([8]). It is not hard to see that a unification problem in *dag*-solved form has a unique most general unifier which can be obtained in a straightforward way [8]. If a set of equations $\mathcal{EQ}$ is in dag-solved form, we say that $\mathcal{EQ}$ is *solved*.

A rewrite rule is an ordered pair (l, r) of terms such that the variables in r also appear in l. It is often written as $l \to r$. A rewrite system $\mathcal{R}$ is set of rewrite rules (l, r). Let $\mathcal{R}$ be a rewrite system and E a set of equations. We define *extended rewriting* with $\mathcal{R}$ modulo E, expressed as

$$s \to_{E \setminus \mathcal{R}} t,$$

if and only if there exist a rule $l \to r$ in $\mathcal{R}$ and a position p in s such that $s|_p \leftrightarrow_E^* \sigma(l),\ t = s\left[\sigma(r)\right]_p$ for some substitution σ. See [9] and [3] for detailed expositions of equational rewriting.

A rewrite rule $l \to r$ is *optimally reducing*[1] if and only if for any substitution θ for which $\theta(r)$ is R-reducible, there is a *proper* subterm s of l such that $\theta(s)$ is R-reducible. A rewrite system R is *optimally reducing* iff every rule in it is optimally reducing modulo R.

3 Unification in $\mathcal{E}_1$

We show that $\mathcal{E}_1$-unification is NP-Complete and give a nondeterministic algorithm for it.

To show NP-Hardness, the NP-complete problem monotone **1-in-3 3SAT** will be polynomially reduced to $\mathcal{E}_1$-unifiability.

The definition of monotone **1-in-3 3SAT** is as follows:

Given: A set of clauses $C = \{c_1, \ldots, c_n\}$ where each clause has exactly three propositional variables.

Question: Is there a satisfying assignment such that exactly one variable is set to *true* in each clause?

Let $C = c_1 \wedge \cdots \wedge c_m$ be an instance of the **1-in-3 3SAT** problem and $V = \{u_1, \ldots, u_n\}$ be variables occurring in C, i.e., $V = Var(C)$. We show how to construct an instance S of the $\mathcal{E}_1$-unification problem from C such that S is unifiable if and only if C is satisfiable.

First of all, we define ground terms a_1, a_2, a_3 as follows:

$$\begin{aligned} a_1 &= B(B(1,0),0) \\ a_2 &= B(B(0,1),0) \\ a_3 &= B(B(0,0),1) \end{aligned}$$

For any clause $c_i = (u_{i_1}, u_{i_2}, u_{i_3})$, $u_{i_j} \in V$, $i = 1, \ldots, m$ and $j = 1, 2, 3$, we introduce a term $T_i = B(B(u_{i_1}, u_{i_2}), u_{i_3})$.

In addition, auxiliary variables x_i, y_i, z_i and a constant m are created. The equation constructed for c_i is

$$U(S(B(m, U(S(B(m, y_i)), a_3))), U(S(B(m, a_1)), T_i)) =^?_{\mathcal{E}_1} U(S(B(m, x_i)), U(S(B(m, z_i)), a_2)).$$

We note that separate variables x_i, y_i, z_i, which are also pairwise distinct, are created for each clause c_i. To follow the results more easily, we define $t_{i,1}$, $t_{i,2}$ and $t_{i,3}$.

$$\begin{aligned} t_{i,1} &= U(S(B(m, a_1)), T_i) \\ t_{i,2} &= U(S(B(m, y_i)), a_3) \\ t_{i,3} &= U(S(B(m, z_i)), a_2) \end{aligned}$$

Therefore, we can now rewrite the equation into the following form:

$$U(S(B(m, t_{i,2})), t_{i,1}) =^?_{\mathcal{E}_1} U(S(B(m, x_i)), t_{i,3})$$

Obviously, we in general obtain a set of equations rather than one equation from a given **1-in-3 3SAT** instance C. Let us denote this set by S.

[1] For term rewriting systems this notion was first introduced in [12], and has been generalized in [6].

Lemma 3.1. *S is unifiable if and only if C is satisfiable.*

Proof. If C is satisfiable, S is unified trivially. Each clause is assigned to one of (1,0,0) or (0,1,0) or (0,0,1). We simply unify corresponding a_k $(k = 1, 2, 3)$ with T_i's in each equation. For instance, if $T_i =^?_{\mathcal{E}_1} a_1$, then $t_{i,1} =^?_{\mathcal{E}_1} S(m)$ and the solution follows from assigning a_3 to y_i and $t_{i,3}$ to x_i. Similar for $T_i =^?_{\mathcal{E}_1} a_2$ and $T_i =^?_{\mathcal{E}_1} a_3$.

Conversely, let S be unified by following the settings above. For each clause c_i in C, it is straightforward to verify that the equation is satisfied if and only if $t_{i,1} =^?_{\mathcal{E}_1} t_{i,2}$ or $t_{i,1} =^?_{\mathcal{E}_1} t_{i,3}$. One of these equations is satisfied when and only when exactly one of $T_i =^?_{\mathcal{E}_1} a_1$ or $T_i =^?_{\mathcal{E}_1} a_2$ or $T_i =^?_{\mathcal{E}_1} a_3$ is unified. By definition, there is only one variable u_{i_j} in T_i assigned to 1 in the solution. We can set the corresponding variable to *true* in each clause of C to build a 1-in-3 satisfying assignment. □

Thus, we finally obtain

Theorem 3.2. *$\mathcal{E}_1$-unification is NP-complete.*

Proof. NP-hardness follows from the previous lemma. Membership in NP follows from the fact that the term rewriting system

$$U(S(B(m, x)), x) \;\rightarrow\; S(m)$$

is optimally reducing and convergent. □

Since unification modulo convergent, optimally reducing term rewriting systems is finitary[2] we get

Theorem 3.3. *$\mathcal{E}_1$-unification is finitary, and there is an algorithm for computing a complete set of $\mathcal{E}_1$-unifiers.*

Proof. An alternative proof would be to observe that $\mathcal{E}_1$ is saturated under ordered paramodulation and then use the result in [10] or [13]. □

However, we also show this in the next section by devising a new algorithm.

3.1 Algorithm

In this section we outline a nondeterministic algorithm for the general $\mathcal{E}_1$-unification problem which we plan to implement. In addition, this algorithm returns a complete set of unifiers for a given problem. We assume, without loss of generality, that each equation is in one of the following standard forms:

1. $X =^? V$
2. $X =^? U(V, Y)$
3. $X =^? B(V, Y)$
4. $X =^? S(V)$
5. $X =^? f(V_1, \ldots, V_n)$

In this setting X, V, $V_1, \ldots, V_n$ and Y are variables and f is an uninterpreted function symbol with arity n.

Transformation rules are created based on the equation forms we specified. Note that rules (h1) and (h2) are nondeterministic and applied "most lazily." The goal is to transform the given set of equations to dag-solved form.

[2]Strictly speaking, this is not shown in [12]. But it is not hard to show, see [6].

(a) $$\frac{\{X =^? V\} \uplus \mathcal{EQ}}{\{X =^? V\} \cup [V/X](\mathcal{EQ})} \quad \text{if } X \text{ occurs in } \mathcal{EQ}$$

(b) $$\frac{\mathcal{EQ} \uplus \{X =^? B(V,Y),\ X =^? B(W,T)\}}{\mathcal{EQ} \cup \{X =^? B(V,Y),\ V =^? W,\ Y =^? T\}}$$

(c) $$\frac{\mathcal{EQ} \uplus \{X =^? S(V),\ X =^? S(W)\}}{\mathcal{EQ} \cup \{X =^? S(V),\ V =^? W\}}$$

(d) $$\frac{\mathcal{EQ} \uplus \{X =^? U(V,Y),\ V =^? S(W'),\ W' =^? B(W,Y))\}}{\mathcal{EQ} \cup \{X =^? S(W),\ V =^? S(W'),\ W' =^? B(W,Y))\}}$$

(e) $$\frac{\mathcal{EQ} \uplus \{X =^? U(V,Y),\ X =^? S(W)\}}{\mathcal{EQ} \cup \{X =^? S(W),\ V =^? S(W'),\ W' =^? B(W,Y))\}}$$

(f) $$\frac{\mathcal{EQ} \uplus \{X =^? U(V,Y),\ X =^? U(W,Y)\}}{\mathcal{EQ} \cup \{V =^? W,\ X =^? U(W,Y)\}}$$

(g) $$\frac{\mathcal{EQ} \uplus \{X =^? U(V,Y),\ X =^? U(V,T)\}}{\mathcal{EQ} \cup \{X =^? U(V,Y),\ Y =^? T\}}$$

(h1) $$\frac{\mathcal{EQ} \uplus \{X =^? U(Y,Z)\}}{\mathcal{EQ} \cup \{Y =^? S(Y'),\ Y' =^? B(M,Z),\ X =^? S(M)\}} \quad \text{if } \mathcal{EQ} \uplus \{X =^? U(Y,Z)\} \text{ is not solved}$$

(h2) $$\frac{\mathcal{EQ} \uplus \{X =^? U(V,W),\ X =^? U(Y,Z)\}}{\mathcal{EQ} \cup \{X =^? U(Y,Z),\ V =^? Y,\ W =^? Z\}}$$

Variables Y', M in rule (h1) and W' in rule (e) are fresh variables.

For uninterpreted function symbols, we have

(i) $$\frac{\mathcal{EQ} \uplus \{X =^? f(V_1,\ldots,V_n),\ X =^? f(W_1,\ldots,W_n)\}}{\mathcal{EQ} \cup \{X =^? f(V_1,\ldots,V_n),\ V_1 =^? W_1,\ \ldots,\ V_n =^? W_n\}}$$

We use rule (a) to eliminate a variable V from the rest of the system. By rules (b), (c), (f), (g) and (i), we remove function symbols from the problem, i.e., narrow the equations. Right after applying those rules, we apply rule (a) to the resulting equations for variable elimination. The soundness of rules (d) – (h2) follow from axiom (1).

Rule (a) is applied most eagerly, followed by the cancellation rules (b), (c), (f), (g) and (i), then (d) and (e) in that order of priority. As mentioned earlier, *the nondeterministic rules (h1) and (h2) have the lowest priority.*

We have the following failure rules:

$$\text{(F1)} \qquad \frac{\mathcal{EQ} \uplus \{X =^? U(V,Y),\ X =^? B(W,T)\}}{FAIL}$$

$$\text{(F2)} \qquad \frac{\mathcal{EQ} \uplus \{X =^? B(V,Y),\ X =^? S(W)\}}{FAIL}$$

We also add a failure rule, which is applied when at least one of f and g is an uninterpreted function symbol.

$$\text{(F3)} \qquad \frac{\mathcal{EQ} \uplus \{X =^? f(V_1,\ldots,V_m),\ X =^? g(W_1,\ldots,W_n)\}}{FAIL} \qquad \text{if } (f \neq g)$$

These rules could be combined into

$$\text{(F4)} \qquad \frac{\mathcal{EQ} \uplus \{X =^? f(V_1,\ldots,V_m),\ X =^? g(W_1,\ldots,W_n)\}}{FAIL} \qquad \text{if } (f \neq g) \text{ and } \{f,g\} \neq \{U,S\}$$

Another kind of failure is occur-check which can be implemented as an extended cycle check as done in algorithms for standard unification. (This can be defined similar to the failure rule (F2) in the next section.) In the presence of the nondeterministic rules (h1) and (h2) this is enough to catch all failures. For instance, consider $X =^? U(Y,X)$. If (h1) is not applied at all, this would cause occur-check failure. But once (h1) is applied we get the set of equations $\{Y =^? S(Y'),\ Y' =^? B(M,X),\ X =^? S(M)\}$ which is unifiable.

Termination can be shown by using the following measure

$$m(S) = (\text{number of occurrences of the symbol } U,\ \text{ number of unsolved variables})$$

The first component decreases in all applications of rules (d) through (h2). Furthermore, it does not increase in rules (a)-(c) and (i). The second component clearly decreases in the case of rule (a); it also decreases for rules (b), (c) and (i), provided that rule (a) is applied immediately afterwards. Since (a) is applied most eagerly, this follows.

Theorem 3.4. *Rules (a)-(i) terminate.*

4 Unification in $\mathcal{E}_2$

We describe an algorithm by using transformation rules, as we did for $\mathcal{E}_1$ in Section 3.1. Furthermore, the algorithm returns a most general unifier if the input equations are unifiable. Without loss of generality, equations will be in one of these forms:

$$X =^? V,\ X =^? B(V,Y),\ X =^? V * Y,\ X =^? f(V_1,\ldots,V_n)$$

(U, V, $V_1,\ldots,V_n$ and Y are variables and f is an uninterpreted function symbol with arity n.)

Both of the function symbols, B and $*$, are cancellative. Note that we do not assume that B or $*$ is commutative or associative.

(a) $$\frac{\{X =^? V\} \uplus \mathcal{EQ}}{\{X =^? V\} \cup [V/X](\mathcal{EQ})} \qquad \text{if } X \text{ occurs in } \mathcal{EQ}$$

(b) $$\frac{\mathcal{EQ} \uplus \{X =^? B(V,Y),\ X =^? B(W,T)\}}{\mathcal{EQ} \cup \{X =^? B(V,Y),\ V =^? W,\ Y =^? T\}}$$

(c) $$\frac{\mathcal{EQ} \uplus \{X =^? V * Y,\ X =^? W * T\}}{\mathcal{EQ} \cup \{X =^? V * Y,\ V =^? W,\ Y =^? T\}}$$

(d) $$\frac{\mathcal{EQ} \uplus \{U =^? B(X,Y),\ U =^? U_1 * U_2\}}{\mathcal{EQ} \cup \{X =^? U_1 * Z,\ U_2 =^? B(Z,Y),\ U =^? U_1 * U_2\}}$$

Rule (d) (the "splitting rule") introduces a fresh variable Z.

To handle uninterpreted functions, we add the same rules as in the case of $\mathcal{E}_1$.

(e) $$\frac{\mathcal{EQ} \uplus \{X =^? f(V_1,\ldots,V_n),\ X =^? f(W_1,\ldots,W_n)\}}{\mathcal{EQ} \cup \{X =^? f(V_1,\ldots,W_n),\ V_1 =^? W_1,\ \ldots,\ V_n =^? W_n\}}$$

A standard failure rule for function clash is:

(F1) $$\frac{\mathcal{EQ} \uplus \{X =^? f(V_1,\ldots,V_m),\ X =^? g(W_1,\ldots,W_n)\}}{FAIL} \qquad \text{if } (f \neq g) \text{ and } \{f,g\} \neq \{B,*\}$$

The outline of the algorithm is as follows: As long as rules are applicable, rules (a) and (F1) are applied most eagerly, and the cancellative rules (b), (c) and (e) come next. The splitting rule (d) is applied, if necessary, at the end, i.e., rule (d) has the lowest priority.

The proof of correctness for this algorithm is similar to the one in Tiden-Arnborg [14].

We define the following relations between terms.

- $U \succ_{r_*} V$ iff there is an equation $U = T * V$
- $U \succ_{l_*} V$ iff there is an equation $U = V * T$
- $U \succ_{r_B} V$ iff there is an equation $U = B(T,V)$
- $U \succ_{l_B} V$ iff there is an equation $U = B(V,T)$
- $U \succ_* V$ iff $U \succ_{r_*} V$ or $U \succ_{l_*} V$
- $U \succ_B V$ iff $U \succ_{r_B} V$ or $U \succ_{l_B} V$
- $U \succ_f V$ iff there is an equation $U = f(\ldots,V,\ldots)$, where f is uninterpreted.

Let $\succ = \succ_{r_*} \cup \succ_{l_*} \cup \succ_{r_B} \cup \succ_{l_B} \cup \succ_f$, i.e., the union of the four relations above. Thus, each of these relations is a subrelation of $\succ$. Alternatively, $\succ = \succ_* \cup \succ_B \cup \succ_f$.

We define an extended occur-check failure rule using $\succ$.

(F2) $\dfrac{\mathcal{EQ}}{FAIL}$ if $X \succ^+ X$ for some X

Let $\sim_{rp(*)}$, and $\sim_{lp(B)}$ be the reflexive, symmetric and transitive closures for $\succ_{r_*}$ and $\succ_{l_B}$, respectively. We also define a set of relations $\beta = \{\beta_1, \beta_2\}$ where

- $\beta_1 = \sim_{rp(*)} \circ \succ_B \circ \sim_{rp(*)}$
- $\beta_2 = \sim_{lp(B)} \circ \succ_* \circ \sim_{lp(B)}$.

One can define two interpretations for $\mathcal{E}_2$, namely interpreting B as left and $*$ as right projections. We denote these interpretations as projection functions symbols $rp(*)$, and $lp(B)$. For instance, if we interpret $*$ as right projection by $rp(*)$, then the axiom $m * B(n, r) = B((m * n), r)$ is trivially satisfied. The same holds if we take B as left projection.

Both interpretations give valid models for the theory. That is, if a problem is solvable modulo $\mathcal{E}_2$, it is also solvable modulo any of these interpretations. This fact is used to prove the following lemma.

Lemma 4.1. *If one of β_1 or β_2 is cyclic, then the problem is not solvable.*

Proof. Without loss of generality assume β_1 is not well-founded. If we interpret $*$ with $rp(*)$ (which gives a model for $\mathcal{E}_2$), it is not hard to see that all variables in the same $\sim_{rp(*)}$-equivalence class become equal to each other. Hence the relation $\succ_B$ becomes not well founded on the set of variables. This implies that there is a cycle w.r.t. $\succ_B$ in the interpreted problem (which is a standard unification problem) and hence there is no solution. Thus the result follows, since the interpreted problem is solvable if the original problem is solvable. A similar argument holds for β_2. □

Therefore, we introduce the following failure rule:

(F3) $\dfrac{\mathcal{EQ}}{FAIL}$ if any of the β_i, $i \in \{1, 2\}$, is cyclic

We will illustrate these with an example. Let $U =^? B(X, Y)$ and $U =^? U_1 * U_2$ be two equations. After rule (d) is applied to this pair of equations, we get

$$U =^? U_1 * U_2,\ X =^? U_1 * Z,\ U_2 =^? B(Z, Y)$$

where Z is a new variable. Now observe that $Z \sim_{lp(B)} U_2$ and $Z \sim_{rp(*)} X$. Thus every new variable introduced by an application of rule (d) is *below* an already existing variables by $\succ_*$ and $\succ_B$ (see Figure 1) and also *equivalent* to existing variables by one of $\{\sim_{rp(*)},\ \sim_{lp(B)}\}$.

Lemma 4.2. *For each equivalence relation in $\{\sim_{rp(*)}, \sim_{lp(B)}\}$, the number of equivalence classes does not increase with the splitting rule.*

Proof. Trivial, since if we apply rule (d), we see that new variable Z 'joins' the existing $\sim_{rp(*)}$- and $\sim_{lp(B)}$-equivalence classes (see Figure 1). □

The number of equivalence classes modulo any equivalence relation will be less than or equal to the number of initial variables in a given problem.

Lemma 4.3. *If rules (a)-(d) are applied infinitely, then one of the relations β_i $(i = 1, 2)$ is cyclic.*

Proof. The only case we need to look at carefully is when the splitting rule is applied. By Lemma 4.2 new variables will not create new equivalence classes; instead they join already existing equivalence classes. Note that for every new variable X created by the splitting rule there is another variable $Y \succ_* X$ which was created earlier. Thus if splitting goes on indefinitely, then we get arbitrarily long chains of the form

$$X_{i_1} \succ_* X_{i_2} \succ_* \dots$$

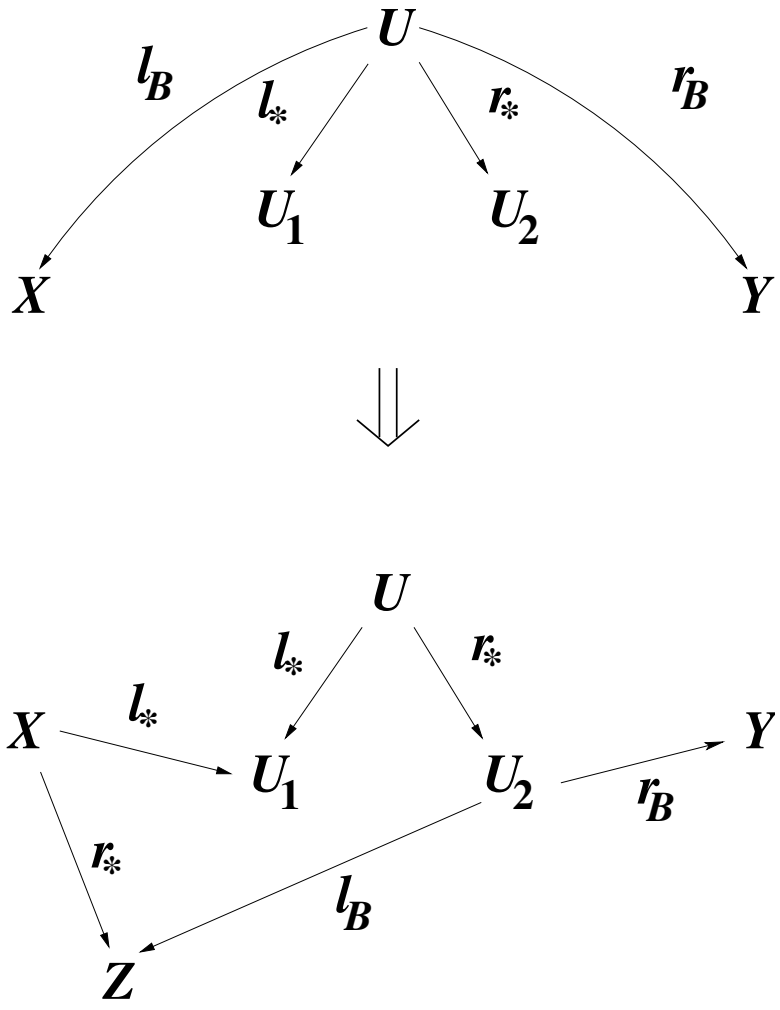

Figure 1: Splitting

But since the number of $\sim_{lp(B)}$- equivalence classes for the problem do not increase, there are indices j and k such that $j < k$ and $X_{i_j} \sim_{lp(B)} X_{i_k}$. (In fact, if n is the number of variables in the original problem, then $j < k \leq n + 1$). This will cause β_2 to be cyclic after finitely many steps. □

Theorem 4.4. *The unification problem modulo $\mathcal{E}_2$ is decidable.*

Proof. Let $\mathcal{S}$ be a problem modulo $\mathcal{E}_2$. If $\mathcal{S}$ is unifiable, rules (a)-(e) will return a solution. On the other hand, if $\mathcal{S}$ is not unifiable, then the possible errors are function clash, occur check error and infinite splitting among variables. Rules (F1) and (F2) detect function clash and occur check in finite time. In the case of infinite splitting, the algorithm will encounter the failure rule (F3) which checks if any of β_i relations is cyclic. Thus, our algorithm decides if $\mathcal{S}$ is unifiable (and computes a comlete set of unifiers) □

Theorem 4.5. *Unifiability modulo $\mathcal{E}_2$ is in P.*

Proof. By Lemma 4.2, we know that the number of equivalence classes remains same throughout the algorithm. Let n be the number of variables. It is easy to see that the number of equivalence classes in both $\sim_{lp(B)}$ and $\sim_{rp(*)}$ is at most n. Note that the algorithm terminates if rule (d) terminates. Rule (d) removes an existing l_B-edge between equivalence classes and adds a new one, which is one level *below the old one* with respect to r_*

(see Figure 1). By Lemma 4.3, this cannot go on for more than n times without failure. Therefore the result follows. □

In the next section, we show that the cardinality of minimal complete set of unifiers is "one".

4.1 Unification Type of $\mathcal{E}_2$

We know that standard forms can be used to represent any problem modulo $\mathcal{E}_2$. Therefore, the transformation rules which we define form a complete method for the problem. We use this fact indirectly to show that the unification type of $\mathcal{E}_2$ is unitary. We first prove that the transformation steps (a)–(e) "preserve unifiers."

Lemma 4.6. *Let $\mathcal{S}$ be a unification problem in standard form and let $\mathcal{S}'$ be obtained after applying one of (a)–(e). Then*

1. *Every unifier of $\mathcal{S}'$ is a unifier of $\mathcal{S}$.*
2. *For every unifier σ of $\mathcal{S}$ there is a unifier $\hat{\sigma}$ of $\mathcal{S}'$ such that $\sigma \equiv_{Var(S)} \hat{\sigma}$.*

Theorem 4.7. *The unification type of $\mathcal{E}_2$ is unitary, and our algorithm computes a complete set of $\mathcal{E}_2$-unifiers.*

Proof. Let $\mathcal{T}$ be the solved form for $\mathcal{S}$. It is easy to see that $\mathcal{T}$ itself is a sequential unifier, and the corresponding parallel unifier, say σ, is a unifier of $\mathcal{S}$. On the other hand, let θ be a unifier of $\mathcal{S}$. By induction on the number of steps we can show, using Lemma 4.6, that there is a unifier $\hat{\theta}$ of $\mathcal{T}$ such that $\theta \equiv_{Var(S)} \hat{\theta}$. We can now show that $\hat{\theta}$ is an instance of σ. □

5 Unification in $\mathcal{E}_1 \cup \mathcal{E}_2$

We show the decidability of unification modulo the union of $\mathcal{E}_1$ and $\mathcal{E}_2$. This theory has the following convergent system:

$$\begin{aligned} U(S(B(m,x)),x) &\rightarrow S(m) \\ m * B(n,r) &\rightarrow B(m*n,r) \end{aligned}$$

Orienting the second axiom the other way causes the Knuth-Bendix completion procedure to diverge.

Let us consider the term rewriting system $U(S(B(m,x)),x) \rightarrow S(m)$ associated with $\mathcal{E}_1$ and denote it by $\mathcal{R}$. Consequently, we use the equational (or class) rewriting relation $\mathcal{R}/\mathcal{E}_2$ and the extended rewriting relation $\mathcal{E}_2\backslash\mathcal{R}$ which is between $\mathcal{R}$ and $\mathcal{R}/\mathcal{E}_2$, i.e.,

$$\mathcal{R} \subseteq \mathcal{E}_2\backslash\mathcal{R} \subseteq \mathcal{R}/\mathcal{E}_2$$

From Section 3, we know that $\mathcal{R}$ is convergent and optimally reducing. We extend these results to $\mathcal{E}_2\backslash\mathcal{R}$. In other words, we show that $\mathcal{E}_2\backslash\mathcal{R}$ is convergent and optimally reducing modulo $\mathcal{E}_2$.

Termination of $\mathcal{R}/\mathcal{E}_2$ follows easily. We observe that for each term t, its equivalence class $[t]_{=_{\mathcal{E}_2}}$ is finite. Applying the rewrite rule in $\mathcal{R}$ modulo $\mathcal{E}_2$ causes a U symbol to disappear. Hence there is no infinite descending chain $t \rightarrow_{\mathcal{R}/\mathcal{E}_2} t' \rightarrow_{\mathcal{R}/\mathcal{E}_2} \ldots$. Furthermore, termination of $\mathcal{E}_2\backslash\mathcal{R}$ follows.

Recall that $\mathcal{R}$ is $\mathcal{E}_2$-confluent if and only if for every s,t such that $s =_{\mathcal{R}\cup\mathcal{E}_2} t$, there exist s',t' such that $s \rightarrow^*_{\mathcal{E}_2\backslash\mathcal{R}} s'$ and $t \rightarrow^*_{\mathcal{E}_2\backslash\mathcal{R}} t'$, and $s' =_{\mathcal{E}_2} t'$ [11].

Lemma 5.1. *$\mathcal{E}_2\backslash\mathcal{R}$ is convergent modulo $\mathcal{E}_2$.*

Proof. This follows from the critical pair criteria given by Jouannaud and Kirchner [9]. That is, if $\mathcal{R}/\mathcal{E}_2$ is terminating and all $\mathcal{E}_2$-classes are finite, then $\mathcal{E}_2\backslash\mathcal{R}$ is confluent if and only if all critical pairs in $CP_{\mathcal{E}_2}(\mathcal{R},\mathcal{R})$ and $CP_{\mathcal{E}_2}(\mathcal{R},\mathcal{E})$ are joinable. Note that all equivalence classes of $\mathcal{E}_2$ are finite and $\mathcal{R}/\mathcal{E}_2$ is terminating. Furthermore, the subterm ordering modulo $\mathcal{E}_2$, denoted as $\succsim_{\mathcal{E}_2}$, is also well-founded [3] since the equation $m * B(n,r) = B(m * n, r)$ is regular and *size-preserving,* i.e., left and right hand sides of the equation are of the same size. Thus the necessary conditions to apply the result in [9] are satisfied. Consider the complete sets of $\mathcal{E}_2$-critical pairs $CP_{\mathcal{E}_2}(\mathcal{R},\mathcal{R})$. The only non-variable position p in term $l = U(S(B(m,x)),x)$ such that $l|_p$ can be unified with (a variant of) l modulo $\mathcal{E}_2$ is $p = \epsilon$ (i.e., at the root). But this leads to a *trivial* critical pair. It is not hard to see that the set of critical pairs $CP_{\mathcal{E}_2}(\mathcal{R},\mathcal{E}_2)$, obtained from overlapping $\mathcal{R}$ "below" $\mathcal{E}_2$, is empty. Since $\mathcal{E}_2$-unification is decidable and unitary, the result follows. □

We next extend the optimal reducibility of $\mathcal{R}$ to the optimal $\mathcal{E}_2$-reducibility of $\mathcal{R}$. A rewrite rule $l \to r$ is *optimally E-reducing* if and only if for any substitution θ for which $\theta(r)$ is $E\backslash R$-reducible, there is a *proper* subterm s of l such that $\theta(s)$ is $E\backslash R$-reducible. A rewrite system R is *optimally E-reducing* iff every rule in it is optimally E-reducing modulo R.

Lemma 5.2. *$\mathcal{R}$ is optimally $\mathcal{E}_2$-reducing.*

Proof. Straightforward, since for all substitutions θ, $\theta(S(m))$ is reducible if and only if $\theta(m)$ is. □

Furthermore, we use the following fact about $\mathcal{E}_2\backslash\mathcal{R}$. Since the root symbol of the left-hand side of $\mathcal{E}_1$, namely U, is not in $Sig(\mathcal{E}_2)$, if s is in normal form and θ is an irreducible substitution modulo $\mathcal{E}_2\backslash\mathcal{R}$, then $\theta(s)$ can be reduced to its $\mathcal{E}_2\backslash\mathcal{R}$-normal form in $|s|$ steps by the innermost reduction strategy.

As mentioned earlier, it was shown by Narendran et al [12] that every optimally reducing and confluent term rewriting system has a decidable unification problem. We give a similar proof to the one in [12] for showing that $\mathcal{E}_2\backslash\mathcal{R}$ is decidable.

Theorem 5.3. *Unification modulo $\mathcal{E}_1 \cup \mathcal{E}_2$ is decidable and finitary, and there is an algorithm to compute a complete set of $\mathcal{E}_1 \cup \mathcal{E}_2$-unifiers.*

Proof. Let s and t be two terms and θ an irreducible substitution that unifies them. $\theta(s)$ and $\theta(t)$ are reduced to $\mathcal{E}_2\backslash\mathcal{R}$-normal forms by the rule $U(S(B(m,x)),x) \to S(m)$ in at most $|s|$ and $|t|$ steps respectively by the innermost reduction strategy. Consider the sequence of positions where the reductions occur. Without performing unification, we instead mimic each reduction step as $s|_p =^?_{\mathcal{E}_2} \eta(U(S(B(m,x)),x))$, where p is a position that a reduction occurs and η an appropriate renaming. Repeat the same for new term $s\,[\eta(S(m))]_p$ and so on. Thus, the idea is to transform the problem by mimicking an innermost reduction sequence where the reductions take place at each *original* term position. We obtain at most $|s| + |t| + 1$ equations to be unified. We apply $\mathcal{E}_2$-unification to the resulting equations and see if there is a solution. Since $\mathcal{E}_2$-unification is decidable and unitary, the result follows. □

6 Conclusion

We have given an equational theory based on the RSA implementation of blind signaures and studied three relevant unification problems. We first considered the two axioms $\mathcal{E}_1$ and $\mathcal{E}_2$ as separate theories and finally unification modulo $\mathcal{E}_1 \cup \mathcal{E}_2$, which turned out to be decidable and finitary. The equational theories we consider are only some of the many possible axiomatizations about blind signatures. Future work would include incorporating other equational axioms. Furthermore, we plan to implement the algorithms and integrate them with the Maude-NPA protocol analyzer [7].

References

[1] S. Anantharaman, H. Lin, C. Lynch, P. Narendran, M. Rusinowitch. "Cap Unification: Application to Protocol Security modulo Homomorphic Encryption". In: *Proc. of the 5th ACM Symp. on Information, Computer and Communications Security*, ASIACCS'10, pp. 192–203, April 2010.

[2] F. Baader, W. Snyder. "Unification Theory". In: *Handbook of Automated Reasoning*, pp. 440–526, Elsevier Sc. Publishers B.V., 2001.

[3] L. Bachmair. *Canonical Equational Proofs.* Birkhäuser 1991.

[4] D. Chaum. "Security without Identification: Transaction System to Make Big Brother Obsolete". *Communications of the ACM* 28(2): 1030–1044, 1985.

[5] D. Chaum. "Blind signatures for untraceable payments". In: *Advances in Cryptology - Crypto '82* 199–203, 1983.

[6] H. Comon-Lundh, S. Delaune. "The finite variant property: how to get rid of some algebraic properties". In: Proc. of *RTA'05* (J. Giesl, ed.), LNCS 3467, pages 294–307. Springer-Verlag, 2005.

[7] S. Escobar, C. Meadows, J. Meseguer. "Maude-NPA: Cryptographic Protocol Analysis Modulo Equational Properties". In: *Foundations of Security Analysis and Design V, FOSAD 2007/2008/2009 Tutorial Lectures* (A. Aldini, G. Barthe, and R. Gorrieri, eds.) LNCS 5705, pages 1–50.

[8] J.-P. Jouannaud, C. Kirchner. "Solving equations in abstract algebras: a rule-based survey of unification." In: *Computational Logic: Essays in Honor of Alan Robinson*, pp. 360–394, MIT Press, Boston (1991).

[9] J.-P. Jouannaud, H. Kirchner. "Completion of a Set of Rules Modulo a Set of Equations". *SIAM J. Comput.* 15(4): 1155–1194, 1986.

[10] C. Lynch, B. Morawska. "Basic Syntactic Mutation." In: *Proc. of CADE 2002* (A. Voronkov, ed.), LNCS 2392, pages 471–485.

[11] C. Meadows, P. Narendran. "A Unification Algorithm for the Group Diffie-Hellman Protocol". In: *Proc. of WITS 2002* 3(1–2): 14–15, 2002.

[12] P. Narendran, F. Pfenning, R. Statman. "On the Unification Problem for Cartesian Closed Categories". *J. Symbolic Logic* 62(2): 636–647, 1997.

[13] R. Nieuwenhuis. "Basic paramodulation and decidable theories." In: *Proc. 11th IEEE Symposium on Logic in Computer Science (LICS'96)* 473–482.

[14] E. Tiden, S. Arnborg. "Unification Problems with One-sided Distributivity". *Journal of Symbolic Computation* 3(1–2): 183–202, 1987.