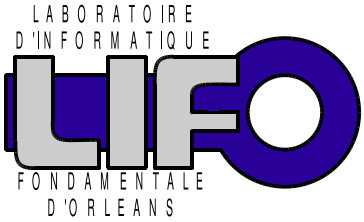


4 rue Léonard de Vinci
BP 6759
F-45067 Orléans Cedex 2
FRANCE
http://www.univ-orleans.fr/lifo

Rapport de Recherche

Equational and Cap Unification for Intrusion Analysis

S. ANANTHARAMAN, LIFO, Orléans (Fr.)
H. LIN, Clarkson University, Potsdam, USA
C. LYNCH, Clarkson University, Potsdam, USA
P. NARENDRAN, University at Albany-SUNY, USA
M. RUSINOWITCH, INRIA-Lorraine, Nancy (Fr.)

Rapport N^o **2008-03**

Equational and Cap Unification for Intrusion Analysis

Siva Anantharaman[1], Hai Lin[2], Christopher Lynch[2], Paliath Narendran[3], and Michael Rusinowitch[4]

[1] Université d'Orléans, France
[2] Clarkson University, Potsdam, NY 13699-5815, USA
[3] University at Albany-SUNY, USA
[4] Loria-INRIA Lorraine, Nancy, France

Abstract. We address the insecurity problem for cryptographic protocols, for an active intruder and a bounded number of sessions. By modeling each protocol step as a rigid Horn clause, and the intruder abilities as an equational theory over a convergent rewrite system, the problem of active intrusion is formulated as a Cap Unification problem. Cap Unification is an extension of Equational Unification: we look for a cap to be placed on a given set of terms, so that it unifies with a given term modulo the equational theory. We give a decision procedure for Cap Unification when the convergent system defining the intruder abilities is dwindling, with some additional assumptions satisfied by usual protocols. We also present briefly a possible way of extending the approach to the (non-dwindling) theory of Homomorphic Encryption (HE) for which passive deduction is known to be decidable. Unification modulo HE is shown to be decidable in the Appendix.

Keywords: Rewriting, Unification, Protocol, Secrecy Analysis.

1 Introduction

Many automated reasoning systems have been designed for representing cryptographic protocols and verifying that they satisfy security properties such as secrecy and authenticity, or to discover bugs. Such systems are often based on model-checking, modal logics, equational reasoning, and resolution theorem-proving (e.g., [17, 3]). Reducing the security problem to a constraint solving problems in a term algebra (modulo an equational theory) is among the most successful approaches: this reduction has proved to be quite effective on standard benchmarks and has also permitted the discovery of new flaws in several protocols (see e.g., [4]).

In particular it is possible to model encryption and decryption operations by collapsing (right-hand sides are variables) convergent rewrite systems, that express simply that decryption cancels encryption, when provided with the right key. Then, extensions of narrowing techniques for semantic unification [12] can be applied to solve the constraints derived from the cryptographic protocol and

the secrecy property that one wants to check. But in general, the procedures for solving protocol insecurity problems modulo rewrite properties of cryptographic operators are involved and/or ad-hoc.

In this paper, we present a novel approach that is simple, in the sense that it is closer to standard unification procedures. Standard equational unification actually turns out to be a particular case of Cap Unification, which is the basis for our inference system for active deduction; we prove that it is sound and complete for dwindling intruder theories, under some minor conditions, met by all usual protocols.

The paper is structured as follows: The preliminaries are presented in Section 2. We introduce, in Section 3, the notions of *Cap Constraints* and *Cap Unification*, and present, in Section 4, the inference system which reduces the active deduction problem to one of solving a set of cap constraints, based on a notion of narrowing with abstraction. The technique is somewhat similar to – but simpler than – those used in [5, 6]. We come up with a decision procedure for solving these cap constraints, by showing its soundness, termination and completeness, for convergent, dwindling, constructor intruder theories under some restrictions. Section 5 presents briefly a possible way of adapting our inference system to the case where the intruder capabilities are modeled as the Homomorphic Encryption theory, which is no longer dwindling, but for which passive deduction is known to be decidable ([1]), as well as equational unification, cf. *Appendix*.

2 Setting the Stage

As usual, Σ will stand for a ranked signature, and $\mathcal{X}$ a countably infinite set of variables. $\mathcal{T} = \mathcal{T}(\Sigma, \mathcal{X})$ is the algebra of terms over this signature; terms in $\mathcal{T}$ will be denoted as $s, t, \ldots$, and variables as $u, v, x, y, z, \ldots$, all with possible suffixes. If f is a member of Σ with at least one argument, then f is a *function symbol*; and if f has no arguments, it is a *constant*. A rewrite rule is a pair of terms (l, r) such that $l \succ r$, for some given reduction ordering $\succ$ on terms; it will be represented as usual, as $l \to r$. A rewrite system R is a finite set of rewrite rules. The notions of reduction and normalization of a term by R are assumed known, as well as those of termination and of confluence of the reduction relation defined by R on terms. R is said to be *convergent* iff the reduction relation it defines on the set of terms is terminating and confluent; R is said to be *dwindling* iff the right hand side (rhs) of any rule in R is a subterm of its left hand side (lhs). Clearly, in a dwindling system R, if the left hand side of a rule in R is reduced by R, then the right hand side must be too.

In this paper, we are concerned with the insecurity problem of protocols, i.e., the problem where a message intended as secret is captured or deduced by an intruder. We model the intruder capabilities as a convergent rewrite system R. In particular, we will consider rewrite systems for which the signature Σ can be divided into two disjoint subsets Σ_D and Σ_C, where $\Sigma = \Sigma_D \cup \Sigma_C$. Symbols in Σ_D are called *defined symbols*, and symbols in Σ_C are called constructors. The system R is called a *constructor system* if the top symbols of all left hand

sides of its rules are defined symbols, and all the other symbols are constructors. The protocol itself is modeled as a set of Horn clauses, referred to as *protocol rules* or *protocol clauses*, that we shall formally define farther down. Protocol insecurity is modeled in two different ways: *passive*, or *active*, deduction. Passive deduction models the intruder knowledge evolution without interaction with the protocol sessions, e.g. via eavesdropping. More precisely, it consists in forming the *cap closure*, in the sense of the following definition – by instantiating SYM as a suitable subset of the symbols in Σ – of a finite set of terms S that models the 'current' intruder knowledge; and adding further terms to this knowledge, via certain R-narrowing steps on the terms of this closure:

Definition 1. *Let S be a given set of terms, and SYM a set of function symbols. Then $Cap(S, SYM)$ is the set of terms defined as follows:*

- $S \subseteq Cap(S, SYM)$
- *If $t_i \in Cap(S, SYM)$, for all $1 \leq i \leq n$, and $f \in SYM$ is of arity n, then $f(t_1, t_2, \ldots, t_n) \in Cap(S, SYM)$.*

An inference system, called saturation of the cap closure, was given in [1] for passive deduction, and was shown to be complete for a class of intruder theories, *strictly* including the class of dwindling theories, and covering in particular the theory of Homomorphic Encryption ("Encryption distributes over pairs").

For modeling active intruder deduction, we need to account for the intruder interactions with the protocol steps. With that purpose, we first model the protocol as a set of *protocol rules* or *protocol clauses* (also called *deduction rules* in the literature); these are defined as follows:

Definition 2. *A* protocol rule *is a pair $(\{t_1, \ldots, t_n\}, t)$ where the t_i's and the t are all terms; it will be denoted as $\{t_1, \ldots, t_n\} \triangleright t$*

Semantics: if σ is a substitution such that the terms $t_i\sigma$, $1 \leq i \leq n$, are already part of the intruder knowledge, then (s)he can deduce the term $t\sigma$.

If R is a given convergent constructor system, and E the associated equational theory, a protocol rule $\{t_1, \ldots, t_n\} \triangleright t$ is said to be an *R- or E-constructed* protocol rule if no function symbol in the rule is a defined symbol of E.

Protocol rules are used to simulate a protocol step in a protocol session. We only consider the analysis of *one protocol session,* since the case of several sessions can be reduced to a single session, via standard techniques [10]. Thus, every protocol rule is used only once; and when the variables of a rule are instantiated, their values are propagated to all the other rules; the variables of a protocol rule are thus considered as rigid variables.

Our next step will be to reduce every protocol clause to a Cap Constraint, and propose a technique of Cap Unification to solve the set of all such constraints.

3 Cap Constraints and Cap Unification

For the developments of this section, R is any given, convergent, rewrite system over some signature Σ, and E the associated equational theory; SYM stands for any given set of symbols in Σ.

For instance, for the Dolev-Yao model with $\Sigma_{DY} = \{p, d, e, \pi_1, \pi_2\}$, where 'p' means pair, 'e' is encryption, 'd' is decryption, 'π_1' (resp. 'π_2') is the projection onto the left (resp. right) component of a pair, the theory E is the one defined by the following convergent, dwindling, constructor system DY:

$$\pi_1(p(x,y)) \to x \qquad d(e(x,y),y) \to x$$
$$\pi_2(p(x,y)) \to y$$

A DY-constructed protocol will *not* contain the symbols d, π_1 and π_2.

Definition 3. *A* cap constraint *is a constraint written in the form $S \rhd_{(SYM,E)} t$, where S is a set of terms, and t is a term. A constraint $S \rhd_{(SYM,E)} t$ is* solvable *iff there exists some $s \in Cap(S, SYM)$ and substitution σ s.t. $s\sigma = t\sigma$ mod E.*

An E-equation (or just 'equation') is, as usual, an equality constraint of the form $s =_E t$, where s and t are (usual) terms. For ease and uniformity of presentation, we agree to identify it with the 'special' cap constraint $s \rhd_{(SYM,E)} t$, whose lhs is now the term s (*not* a set of terms); if we also agree to set $Cap(s, SYM) = \{s\}$, then obviously solving the special cap constraint reduces to unifying s and t mod E.

Definition 4. *Let $\{S_i \rhd_{(SYM,E)} t_i,\ 1 \le i \le n\}$, be any given set of cap constraints (some may be special). A substitution σ is a* solution *to this set iff:*

- *σ solves each cap constraint $S_i \rhd_{(SYM,E)} t_i$, $1 \le i \le n$.*
- *Suppose a variable x is in the rhs of the constraints: $S_{i_k} \rhd t_{i_k}$, $1 \le k \le r$. Then $x\sigma \in Cap(S_{i_k}\sigma, SYM)$ for* every *$k \in \{1, \ldots, r\}$.*

The Reduction to Cap Constraints: We show here how to generate a set of cap constraints from the protocol rules such that an intruder can deduce some message m iff the set of cap constraints is solvable. We use a simple example to illustrate the idea. Suppose we are given the following set of protocol rules:

$$\{s_1, s_2\} \rhd t_1$$
$$\{s_3\} \rhd t_2$$
$$\{s_4\} \rhd t_3$$

In these protocol rules, each s_i and t_j is a term, possibly containing variables, and as we said before each occurrence of any variable cannot be instantiated more than once. The intruder's initial knowledge is $\{a, b\}$, and the goal is to check if an intruder can deduce the secret message m. First, we nondeterministically guess a sequence of numbers; for example, 1 and 3, meaning the first then the third protocol rules are used by the intruder. Then the following set of cap constraints are generated (where Σ is a superset of the symbols appearing in E):

$$\{a, b\} \rhd_{(\Sigma,E)} s_1$$
$$\{a, b\} \rhd_{(\Sigma,E)} s_2$$
$$\{a, b, t_1\} \rhd_{(\Sigma,E)} s_4$$
$$\{a, b, t_1, t_3\} \rhd_{(\Sigma,E)} m$$

The lhs of each constraint indicates the knowledge of the intruder at some given instant, and the rhs the knowledge that (s)he can deduce from the protocol step. For all sets $\mathcal{S}$ of cap constraints that we will consider in this paper, there will be a way of ordering its constraints, such that:

- The intruder loses no information, at any instant.
- Any variable x that appears in the lhs of some constraint, must also appear in the rhs of a preceding constraint in $\mathcal{S}$.

(cf. Definition 8). In particular, this will imply that the terms representing the intruder's initial knowledge can only be ground.

3.1 Cap Unification - Empty Theory

Our objective is to give an algorithm for solving cap constraints over some specific equational theories. Before doing that, we first give an algorithm for solving cap constraints in the empty theory. This algorithm *Cap Unification* will be a generalization of the standard unification algorithm for syntactic unification, that accounts for the caps on terms. In order to solve one cap constraint modulo the empty theory, we will transform it to a set of cap constraints and equations. In the algorithm given below, Γ represents (to the rest of this subsection) a set of cap constraints and (usual) equations modulo the empty theory.

We first formulate the 'standard' inference rules dealing with usual unification, We refer to these rules, respectively, as **Trivial**, **Noncap Decomposition**, **Orient**, **Variable Substitution**, **Clash**, and **Occur Check**. The last two rules are to detect failure.

1. $\Gamma \sqcup \{t = t\} \Rightarrow \Gamma$
2. $\Gamma \sqcup \{f(s_1, s_2, \ldots, s_m) = f(t_1, t_2, \ldots, t_m)\} \Rightarrow$
   $\Gamma \sqcup \{s_1 = t_1\} \sqcup \{s_2 = t_2\} \sqcup \ldots \sqcup \{s_m = t_m\}$
3. $\Gamma \sqcup \{t = x\} \Rightarrow \Gamma \sqcup \{x = t\}$
   if x is a variable and t is not a variable.
4. $\Gamma \sqcup \{x = t\} \Rightarrow \Gamma\sigma \sqcup \{x = t\}$
   if x is not a proper subterm of t, x occurs in Γ, σ is the substitution $[x \mapsto t]$.
5. $\Gamma \sqcup \{f(s_1, s_2, \ldots, s_m) = g(t_1, t_2, \ldots, t_m)\} \Rightarrow fail$
   if $f \neq g$ (both f and g can be a function symbol or a constant).
6. $\Gamma \sqcup \{x = t\} \Rightarrow fail$
   if x is a proper subterm of t.

The additional inference rules for cap unification are defined as transformations of sets of cap constraints and equations, for any set SYM of function symbols, and over some theory: E.

The simplest case would be: One of the elements in S is unifiable with s, and this gives the following **Decomposition** rule:

– $\Gamma \sqcup \{S \rhd_{(SYM,E)} f(t_1, t_2, \ldots, t_m)\} \Rightarrow \Gamma \sqcup \{s_i = f(t_1, t_2, \ldots, t_m)\}$, $1 \leq i \leq n$.

We use the following **Cap Decomposition** rule to take care of the fact that the intruder is able to build up terms using any symbol in SYM:

- $\Gamma \sqcup \{S \rhd_{(SYM,E)} f(t_1, t_2, \ldots, t_m)\} \Rightarrow$
  $\Gamma \sqcup \{S \rhd_{(SYM,E)} t_1\} \sqcup \{S \rhd_{(SYM,E)} t_2\} \sqcup \ldots \sqcup \{S \rhd_{(SYM,E)} t_m\}$, if $f \in SYM$.

The two decomposition rules allow us to break down a cap constraint of the form $S \rhd_{(SYM,E)} t$ to a (possibly empty) set of cap constraints, all of the form $S \rhd_{(SYM,E)} x_1 \sqcup \cdots \sqcup S \rhd_{(SYM,E)} x_n$. In other words, the constraints resulting from this process contain the same set on the left hand side as the original set, and they each contain a single variable on the right hand side.

Suppose we have $S \rhd_{(SYM,E)} x$, then all we can say about x is that $x \in Cap(S, SYM)$, then we create an equation $x = S$; so we have the following **Create Set Equation** rule:

- $\Gamma \sqcup \{S \rhd_{(SYM,E)} x\} \Rightarrow \Gamma \sqcup \{x = S\}$

This leads us to a definition:

Definition 5. *$\{t_1, t_2, \ldots, t_n\}$ is a set-term iff $t_1, t_2, \ldots, t_n$ are (usual) terms.*

So we have two kinds of terms: usual terms and set-terms, in the last three inference rules. But *set-terms are allowed only to the left of cap constraints, and to the right of equality constraints whose lhs are variables*. We allow variables to be substituted by set-terms. For example, we could apply $x = \{t_1, t_2\}$ to $f(x)$, and the result is $f(\{t_1, t_2\})$. which is an abbreviation for the set-term $\{f(t) | t \in Cap(\{t_1, t_2\}, SYM)\}$. Similar to usual terms, a set-term is ground iff there is no variable in it.

The *Cap Unification* procedure is defined by *all* the above rules. As usual, we write: $\Gamma \Rightarrow \Gamma$' if the constraint set Γ' is derived from the constraint set Γ by applying one of the inferences. In a set of equality constraints, a variable x is said to be *solved* iff x appears *only once*, and as the lhs of an equation of the form $x = t$ with t a term, or set-term. We define a *normal form* Γ of cap unification to be a set of equalities $\{x_1 = t_1, \cdots, x_n = t_n, y_1 = S_1, \cdots, y_m = S_m\}$, where each x_i is a solved variable, each t_i is a *usual term*, each y_j is a variable, and each S_j is a set-term. Note that the y_j variables are allowed to be repeated.

Just like usual syntactic unification, cap unification (over the empty theory) always terminates:

Lemma 1. *There is no infinite chain of cap unification inferences.*

Proof. Given a set of cap constraints Γ, define a measure $\mu(\Gamma)$ as the 4-tuple (p, q, r, s), where p is the number of unsolved variables in Γ, q is the number of symbols in Γ, r is the number of ("reversed") equations in Γ of the form $t = x$ where x is a variable, and s is the number of constraints in Γ of the form $S \rhd x$ where x is a variable. The ordering, with p, q, r, s compared lexicographically, is well-founded. If $\Gamma_1 \Rightarrow \Gamma_2$ then $\mu(\Gamma_1) > \mu(\Gamma_2)$: This is because Variable Substitution reduces the first component of the measure; Orient reduces the third and

does not increase the others; Create Set Equation reduces the fourth and does not increase any others; all the others decrease the second component and do not increase the first. □

Cap unification is sound (by inspection of the transformation rules):

Lemma 2. *Let σ be a (usual) term substitution. If $\Gamma_1 \Rightarrow \Gamma_2$ and σ is a solution for Γ_2, then σ is also a solution for Γ_1.*

Cap unification is complete:

Lemma 3. *Let σ be a (usual) term substitution. If σ is a solution of Γ_1, then there exists a Γ_2 such that $\Gamma_1 \Rightarrow \Gamma_2$ and σ is a solution of Γ_2.*

Proof. Consider any unsolved constraint c in Γ_1. If c is a cap constraint of the form $S \triangleright t$ where t is not a variable, then a suitable Noncap or Cap Decomposition rule can be applied to Γ_1, so as to preserve the solution. If the rhs t of the constraint c *is* a variable, then only the Make Set Equation rule is applicable, and this obviously preserves the solution. Similarly, in the case where c is an equation $s = t$, where s is a usual term that is not a solved variable, some transformation applies and preserves the solution. □

So we have shown that a satisfiable Cap unification problem can always be reduced to an equivalent normal form, and an unsatisfiable one cannot be reduced to normal form. Given a satisfiable set of constraints Γ, let $nf(\Gamma) = \{x_1 = t_1, \cdots, x_n = t_n, y_1 = S_1, \cdots, y_m = S_m\}$ be a normal form for Γ, produced by Cap unification. We then define a notion of $mgcu(nf(\Gamma))$ as follows. Let σ_0 be a most general solution for the part $\{x_1 = t_1, \cdots, x_n = t_n\}$ formed of the usual equalities in $nf(\Gamma)$; then σ_0 is first extended as a substitution σ' assigning set-terms to the other variables y_j in $nf(\Gamma)$, under the following condition:

- For every $j, 1 \leq j \leq m$, $y_j\sigma' = S'_j$ is the smallest (non-empty) sub-set-term of $S_j\sigma_0$ such that $S'_j \subset Cap(S_i\sigma_0, SYM)$ for *every* i with $y_i = y_j$.

Since the underlying equational theory is empty, and since the set-terms considered are all finite, this is a finitary notion. We then set σ to be the set-term substitution obtained by fully developing σ' (i.e., fully instantiating all the variables in its range), and define: $mgcu(\Gamma) = mgcu(nf(\Gamma)) = \sigma$. Such a substitution is said to be ground iff its range has no variables. Note that, when all the S_i's are single terms, *mgcu* coincides with *mgu*.

In addition, for some of the proofs of Section 4.2, we will be needing the notion of a *restricted solution* for Γ:

Definition 6. *A* restricted solution *for a set of constraints in normal form $\Gamma = \{x_1 = t_1, \cdots, x_n = t_n, y_1 = S_1, \cdots, y_m = S_m\}$ is a substitution σ of the form $[x_1 \mapsto t_1, \cdots, x_n \mapsto t_n, y_j \mapsto t, 1 \leq j \leq m]$, where the $t_1, \cdots, t_n$ are usual ground terms, and t is a ground term in $\bigcap_{1 \leq j \leq m} S_j$.*

4 Inference System for the Dwindling Case

In this section, we extend the Cap Unification inference system over the empty theory, given in the preceding section, by adding a set of inference rules for solving cap constraints modulo a non-empty equational theory E defined by a convergent, dwindling, constructor system R, under typical conditions met by all usual protocols. We will show that this extended inference system is sound and complete for such theories. *For the rest of the section, SYM will stand for* $sig(E)$, the set of all symbols in the signature of E.

In these inference rules, any constraint set (in the premises or conclusions) is divided into a cap constraint part denoted Γ, and an equality constraint part denoted C. We represent it in the form $\Gamma[\![C]\!]$. We apply our inferences to the cap constraint part, and the equational part will represent the results of solving the cap unification problems in the inferences. Initially, the equational part will be the empty. (This is similar to Basic Narrowing, where the unification problems are saved instead of being applied to the narrowing problem.) As above, set-terms may appear to the left of cap constraints, and to the right of equality constraints whose lhs are variables. Before introducing our inference rules, we need the following definition.

Definition 7. *We define a function* $SetUsualTerms(t)$ *inductively as follows:*
$SetUsualTerms(t) = \{t\}$, *if* t *is a usual term.*
$SetUsualTerms(t) = t$ *if* $t = \{t_1, \ldots, t_n\}$ *is a set-term, i.e. a set of usual terms.*
$SetUsualTerms(t) = SetUsualTerm(t_1) \cup \cdots \cup ToUsualTerms(t_n)$,
if $t = f(t_1, t_2, \ldots, t_n)$ *and some subterm of* t *is a set-term.*

The inference system will be denoted as $\mathcal{I}_D$. In addition to the inference rules above, for Cap Unification, it will contain the following inference rules:

$$\frac{\Gamma \sqcup \{S \rhd_{(SYM,E)} t\}[\![C_1]\!]}{\Gamma \sqcup \{S \sqcup \{x\} \rhd_{(SYM,E)} t\}[\![nf(C_1 \sqcup \{x = t'\} \sqcup \{S \rhd_{(SYM,\emptyset)} s'\})]\!]} \text{ (Cap Narrowing)}$$

if: (i) $s' \to t' \in R$; (ii) There exists a $v' \in SetUsualTerms(t'\sigma)$ s.t. v' is not a superterm of any $v \in SetUsualTerms(s\sigma)$, for $s \in S$, and $\sigma = mgcu(S \rhd s')$; and (iii) For every variable $y \in S$, (y)mgcu(C_1) is ground. (The x in the conclusion is a *fresh variable.*)

$$\frac{\Gamma \sqcup \{S \rhd_{(SYM,E)} t\} \ [\![C_1]\!]}{\Gamma \ [\![nf(C_1 \sqcup \{S \rhd_{(SYM,\emptyset)} t\})]\!]} \text{ (Cap Unification)}$$

if S and t are cap unifiable (over the empty theory).

The following rule is essential for termination, thus is used eagerly.

$$\frac{\Gamma \sqcup \{S \sqcup \{x\} \rhd_{(SYM,E)} t\}[\![C_1]\!]}{\Gamma \sqcup \{S \rhd_{(SYM,E)} t\}[\![C_1]\!]} \text{ (Variable Elimination)}$$

if some equality constraint $x = T$ is already present in C_1.

We often write an inference as a transformation $P \Rightarrow_{\mathcal{I}_D} C$ where P is the premise and C is the conclusion. The inference rules are applied nondeterministically, except for Variable Elimination which is applied eagerly: meaning it must be applied whenever applicable. We must show that the rules terminate and are sound; and also show that they are complete, i.e., that if σ solves a constraint Γ then that there is some derivation from Γ to $\emptyset[\![C]\!]$, where C is satisfiable.

Example. To give an example of our inference system, suppose that E contains the rule $d(e(h(z), y), y) \rightarrow h(z)$, and suppose that S is the set of terms $\{e(h(a), b), b\}$. We want to show that $\{S \rhd_{(SYM,E)} h(a)\}$. This can be solved by a sequence of two inferences. The first inference is a Cap Narrowing inference, whose result is $\{S \sqcup \{x\} \rhd_{(SYM,E)} h(a)\} \quad [\![\{x = h(a), z = a, y = b\}]\!]$. The set $\{x = h(a), z = a, y = b\}$ is the normal form of $\{x = h(z), S \rhd_{(SYM,\emptyset)} d(e(h(z), y), y)\}$, which is obtained by Cap Unification. The steps for solving Cap Unification are as follows. First Cap Decomposition is applied, resulting in $\{x = h(z), S \rhd_{(SYM,\emptyset)} e(h(z), y), S \rhd_{(SYM,\emptyset)} y\}$. Next, Decomposition gives us $\{x = h(z), h(a) = h(z), b = y, S \rhd_{(SYM,\emptyset)} y\}$. After a few standard unification steps, we get $\{x = h(a), z = a, y = b, S \rhd_{(SYM,\emptyset)} b\}$. Finally, one more Decomposition step gives us $\{x = h(a), z = a, y = b\}$. This Cap Narrowing Step is applicable because $h(a)$ is not a superterm of any term in S, so condition (ii) applies; and all the variables in S are ground so condition (iii) also applies. After the Cap Narrowing Step, one inference of Cap Unification gives us $\emptyset \quad [\![\{x = h(a), z = a, y = b\}]\!]$, and the problem is now solved.

4.1 Termination

For termination and for completeness, there are certain typical protocol properties that will be required. The properties that we formulate now, are true for standard protocols; they will be preserved under the inference rules of $\mathcal{I}_D$.

Definition 8. *Let* $\Gamma = \{S_1 \rhd_{(SYM,E)} t_1, \cdots, S_m \rhd_{(SYM,E)} t_m\}[\![C]\!]$ *be an* ordered *set of cap constraints.*

1. *The* No Information Loss *property: Let* $1 \leq i < j \leq m$*, and* σ *any ground substitution. If* $t_i \in Cap(S_i\sigma, SYM)$ *then there exists a* $t'_j \in Cap(S_j\sigma, SYM)$*, such that* $t_i = t'_j$ *modulo* E*.*
2. *The* Variable Introduction *property: Let* $1 \leq j \leq m$*,* $\sigma = mgcu(C)$*, and* x *any variable appearing in* $S_j\sigma$*; then there exists* $i, 1 \leq i < j$*, such that* x *is already in* $t_i\sigma$*.*
3. *The* Nonempty Knowledge *property:* S_1 *contains some ground constant.*
4. *The* Constructor *property: No* defined symbol *of* R *appears in* Γ*.*
5. *The* Derivable Set *property: Let* $1 \leq i, j \leq m$ *be such that* $x \in S_j$*, and the constraint* C *contains the equality* $x = S_i$*; then all the* $t \in S_i$ *are also in* $Cap(S_j - \{x\}, SYM)$*.*

Any given constraint set Γ *is said to have these properties iff its cap constraints can be ordered in some way to satisfy all these properties. Together, these properties are called the* Standard Protocol Property *for* Γ*.*

We assume that the initial cap constraint sets derived from protocol rules have the Standard Protocol Property: No information loss means that once the intruder knows something, (s)he will not forget it. Variable introduction means that a principal's actions are determined by the messages it receives or deduces. Nonempty knowledge means that the intruder knows something. The constructor property says that the protocol clauses do not contain functions that destruct data. The derivable set property is trivially true, because initially there are only cap constraints (no equality constraints).

The inference process is modeled by $\mathcal{I}_D$-derivation sequences:

Definition 9. *Let $\Gamma_0[\![C_0]\!]$ be a set of cap constraints. An $\mathcal{I}_D$-derivation sequence from $\Gamma_0[\![C_0]\!]$, is a sequence of the form $\Gamma_0[\![C_0]\!], \Gamma_1[\![C_1]\!], \ldots, \Gamma_n[\![C_n]\!]$ where, for all $1 \leq i \leq n$, $\Gamma_i[\![C_i]\!]$ is obtained from $\Gamma_{i-1}[\![C_{i-1}]\!]$ by an inference rule of $\mathcal{I}_D$.*

Our purpose in this subsection is to show the termination of any $\mathcal{I}_D$-derivation. For that we need to show that $\mathcal{I}_D$-derivation sequences preserve the Standard Protocol Property; and for this, we shall assume as we may, that if $\Gamma[\![C]\!]$ is an ordered set of cap constraints, and if $\Gamma[\![C]\!] \Rightarrow_{\mathcal{I}_D} \Gamma'[\![C']\!]$, *then the latter constraint set is ordered 'in the same way' as the former* (any new inferred constraint takes the place of the constraint on which the inference is performed).

Lemma 4. *Let $\Gamma_0[\![\emptyset]\!], \Gamma_1[\![C_1]\!], \ldots, \Gamma_n[\![C_n]\!]$ be an $\mathcal{I}_D$-derivation, where $\Gamma_0[\![\emptyset]\!]$ has the Variable Introduction property. Then $\Gamma_n[\![C_n]\!]$ also has the Variable Introduction property.*

Proof. We will show that if $\Gamma[\![C]\!] \Rightarrow_{\mathcal{I}_D} \Gamma'[\![C']\!]$, and $\Gamma[\![C]\!]$ has the Variable Introduction property, then so does $\Gamma'[\![C']\!]$.

Let $\sigma = mgcu(C)$, and $\sigma' = mgcu(C')$. Suppose $S'_j \triangleright t'_j$ is a constraint in Γ', and suppose S'_j contains some term s' such that a variable x appears in $s'\sigma'$. Now, the constraint $S'_j \triangleright t'_j$ must come from some $S_j \triangleright t_j$ in Γ.

If the term s' appears already in S_j then there is a variable y in $s'\sigma$ such that $y\sigma'$ contains x. By assumption, $\Gamma[\![C]\!]$ has the Variable Introduction property, so there is a $1 \leq k < j$ and a cap constraint $S_k \triangleright t_k$ in Γ such that y is already in $t_k\sigma$. But then there is a corresponding $S'_k \triangleright t'_k$ in Γ' with x appearing in $t_k\sigma'$.

The only other case – that need not be considered – would be that x is a new variable created by Cap Narrowing: in that case, $x\sigma'$ would be ground. □

Corollary 1. *Let $\Gamma_0[\![\emptyset]\!], \Gamma_1[\![C_1]\!], \ldots, \Gamma_n[\![C_n]\!]$ be an $\mathcal{I}_D$-derivation. Let for any $i, 0 \leq i \leq n$, $\tau_i = mgcu(C_i)$; then $\Gamma_i\tau_i$ is ground (i.e., all of its terms and set-terms are ground).*

Proof. Follows from the above lemma, and the fact that the first cap constraint in the ordering can only have ground terms to the left. □

Lemma 5. *Let $\Gamma_0[\![\emptyset]\!], \Gamma_1[\![C_1]\!], \ldots, \Gamma_n[\![C_n]\!]$ be an $\mathcal{I}_D$-derivation, where $\Gamma_0[\![\emptyset]\!]$ has the No Information Loss and Derivable Set properties. Then $\Gamma_n[\![C_n]\!]$ also has the No Information Loss and the Derivable Set properties.*

Proof. We will show that if $\Gamma_i[\![C_i]\!]$ has the No Information loss and Derivable Set property, then so does $\Gamma_{i+1}[\![C_{i+1}]\!]$.

By definition, Cap Narrowing and Variable Elimination preserve the No information Loss. Cap Unification also preserves it, because $\Gamma_i[\![C_i]\!]$ has the Derivable Set property.

We now have to show that $\Gamma_{i+1}[\![C_{i+1}]\!]$ also has the Derivable Set property. Consider then a variable x, and an equality $x = t$ – where t is a set-term or term – that appears in C_{i+1}. If this equality appears already in C_i, then the fact that $\Gamma_i[\![C_i]\!]$ has the Derivable Set property implies that $\Gamma_{i+1}[\![C_{i+1}]\!]$ must also. On the other hand, if the equality $x = t$ does not appear in C_i, that means it got created in C_{i+1}, by a Cap Narrowing or Cap Unification inference; but then, t must have appeared in some cap constraint in the constraint set Γ_i where x was some t'. Because of the No Information Loss property, this implies the Derivable Set property for $\Gamma_{i+1}[\![C_{i+1}]\!]$. □

Lemma 6. *Let $\Gamma_0[\![\emptyset]\!], \Gamma_1[\![C_1]\!], \ldots, \Gamma_n[\![C_n]\!]$ be an $\mathcal{I}_D$-derivation, where $\Gamma_0[\![\emptyset]\!]$ has the Nonempty Knowledge property. Then $\Gamma_n[\![C_n]\!]$ also has the Nonempty Knowledge property.*

Proof. We have to show that if $\Gamma_i[\![C_i]\!]$ has the Nonempty knowledge property, then so does $\Gamma_{i+1}[\![C_{i+1}]\!]$. The only problem would be if an inference made the initial knowledge given by the ground constants of Γ_0 disappear. The No Information Loss property implies that this is not so. □

Lemma 7. *Let E be a constructor system. Suppose $\Gamma_0[\![\emptyset]\!], \Gamma_1[\![C_1]\!], \ldots, \Gamma_n[\![C_n]\!]$ is an $\mathcal{I}_D$-derivation, where $\Gamma_0[\![\emptyset]\!]$ has the Constructor property. Then $\Gamma_n[\![C_n]\!]$ also has the Constructor property.*

Proof. We have to show that if $\Gamma_i[\![C_i]\!]$ has the Constructor property, then so does $\Gamma_{i+1}[\![C_{i+1}]\!]$. But, because the rules in the system R are all constructor rules, there is no way to create a new defined symbol in an inference. □

Proving Termination: We need to find a well-founded measure which is decreased by the inferences. For defining a termination measure, and for some of the proof details to come, it will be convenient to have a notion of cap-term:

Definition 10. *A* cap-term *is defined inductively as follows:*
(i) S is a cap-term if S is a set-term.
(ii) $f(s_1, s_2, \ldots, s_n)$ is a cap-term if, $f \in SYM$, and for $1 \leq i \leq m$, s_i is either a cap-term, or $s_i \in s_j$, for some $1 \leq j \leq m$, $j \neq i$.

Let t be any 'term' possibly containing cap-subterms. A cap-term is said to be *maximal* in t if none of its superterms in t is a cap-term; in other words, any superterm of a maximal cap-term in t has at least one usual term as argument or subterm. For defining our measure, we will be using a transformation function on 'terms' possibly containing cap-subterms, that we define as follows:

Let $\mathbf{w}$ denote a fixed *new* constant. For any 'term' t possibly containing a cap-subterm, $\mathbf{w}(t)$ will denote the usual term obtained by replacing all maximal

cap-subterms of t by $\mathbf{w}$. This function is extended, in a natural manner, to cap constraints and sets of cap constraints. Based on this function, we now define two functions K_{pos} and T for any cap constraint set $\Gamma[\![C]\!]$, as follows:

$K_{pos}(\Gamma[\![C]\!])$ is the number of non-variable positions in the cap constraint part of Γ.

Given any set of usual terms $\mathcal{S}$, let $Sub(\mathcal{S})$ denote the set of all usual subterms of terms in $\mathcal{S}$, and $Diff(\mathcal{S}) = Sub(\mathcal{S}) \setminus \mathcal{S}$. We then set:

$T(\Gamma[\![C]\!])$ is the multiset $\{Diff(S\sigma) \mid S \triangleright t \in \Gamma, \sigma = mgu(\mathbf{w}(C))\}$

We define now the measure as: $M(\Gamma[\![C]\!]) = (K_{pos}(\Gamma[\![C]\!]), T(\Gamma[\![C]\!]))$, which is compared lexicographically. This measure is well-founded. We show that it is reduced by each inference:

Lemma 8. *Suppose $\Gamma[\![C]\!]$ has the Standard Protocol property. If $\Gamma[\![C]\!] \Rightarrow_{\mathcal{I}_D} \Gamma'[\![C']\!]$, then $M(\Gamma'[\![C']\!]) < M(\Gamma[\![C]\!])$.*

Proof. For Cap Unification, the first component of the measure always decreases. Since the Variable Elimination inference rule is applied eagerly, there will be no variable in the set on the left hand side of a cap constraint. So the number of nonvariable positions in S is decreased by Cap Unification.

For Variable Elimination, we cannot increase the first component of the measure, but the second measure is decreased.

Suppose we do Cap Narrowing on some constraint $S_i \triangleright t_i$ and $s' \to t'$, $\sigma = mgcu(S_i \triangleright s')$. We consider $s'\sigma$. There are three cases.

Case 1: Suppose $s'\sigma$ is some ground term t, where no subterm of t is a set-term. E being dwindling, the result of Cap Narrowing can only be a subterm, otherwise, the result is already in $Cap(S, SYM)$; so the inference is not performed.

Case 2: Suppose $s'\sigma$ is some ground cap-term t. According to the definition of cap-term. For all $t' \in SetUsualTerms(t)$, there exists some S_j s.t. $j \leq i$ and $t' \in SetUsualTerms(S_j)$. Because of the No Information Loss Property, $t' \in SetUsualTerms(S_i)$. Therefore, Cap Narrowing is not allowed because of the second condition for Cap Narrowing.

Case 3: Suppose $s'\sigma$ is a ground term $u[t]$, where t is a a cap-term, and $u[t]$ is not a cap-term. Let $\mathbf{w}(v)$ be the function (defined above) that replaces every maximal cap-subterm in any 'term' v by the new constant $\mathbf{w}$. Then $\mathbf{w}(u[t]) = u[\mathbf{w}]$, and the result of Cap Narrowing would be $u'[\mathbf{w}]$, a subterm of $u[\mathbf{w}]$. □

Theorem 1. *Suppose $\Gamma[\![\emptyset]\!]$ is a constraint set satisfying the Standard Protocol property. Then every $\mathcal{I}_D$-derivation sequence from $\Gamma[\![\emptyset]\!]$ is finite.*

Proof. The measure M decreases at each inference, and M is well-founded. □

4.2 $\mathcal{I}_D$ is Sound and Complete

In this section, we prove that our inference system $\mathcal{I}_D$ is sound and complete for convergent, dwindling, constructor systems). We thus get - in combination with termination proved above - a decision procedure for solving Cap Unification modulo such systems. We first prove soundness of $\mathcal{I}_D$.

Theorem 2. *Let $\Gamma_0[\![\emptyset]\!]$ be a constraint set satisfying the Standard Protocol property. Suppose $\Gamma_0[\![\emptyset]\!], \Gamma_1[\![C_1]\!], \ldots, \Gamma_n[\![C_n]\!]$ is an $\mathcal{I}_D$-derivation, and σ a restricted ground substitution (cf. Definition 6) that satisfies $\Gamma_n[\![C_n]\!]$; then σ satisfies Γ_0.*

Proof. Assuming σ satisfies $\Gamma_{i+1}[\![C_{i+1}]\!]$, we prove that σ also satisfies $\Gamma_i[\![C_i]\!]$. This is true, since no inference rule, from step i to step $i+1$, adds any constraint that can be inconsistent with $\Gamma_i[\![C_i]\!]$. □

We now prove completeness of $\mathcal{I}_D$.

Theorem 3. *Let Γ_0 be a set of cap constraints, $C_0 = \emptyset$, and suppose $\Gamma_0[\![C_0]\!]$ satisfies the Standard Protocol property. Let σ be a restricted ground substitution satisfying Γ_0. Then there is an $\mathcal{I}_D$-derivation sequence $\Gamma_0[\![C_0]\!], \ldots, \Gamma_n[\![C_n]\!]$ such that $\Gamma_n = \emptyset$ and σ satisfies C_n.*

Proof. We assume that σ satisfies $\Gamma_i[\![C_i]\!]$ with $\Gamma_i \neq \emptyset$ and show that there is an inference step $\Gamma_i[\![C_i]\!] \Rightarrow_{\mathcal{I}_D} \Gamma_{i+1}[\![C_{i+1}]\!]$ such that σ satisfies $\Gamma_{i+1}[\![C_{i+1}]\!]$.

Since σ satisfies $\Gamma_i[\![C_i]\!]$, then for each of its cap constraints $S \rhd_{(SYM,E)} t$, we know that there is some $s \in Cap(S, SYM)$ with $s\sigma = t\sigma$ modulo R. This means that $t\sigma$ has an R-rewrite proof to $s\sigma\downarrow_R$. Let $S \rhd_{(SYM,E)} t$ be the first constraint in $\Gamma_i[\![C_i]\!]$ for the ordering from the Standard Protocol property. Since $\Gamma_i[\![C_i]\!]$ obeys the Variable Introduction property, we know that $\Gamma_i\tau$ must be ground, where $\tau = mgcu(C_i)$, cf. Corollary 1.

Suppose there are no steps in the rewrite proof: meaning $t\sigma$ is identical to $s\sigma$; then there is a Cap Unification inference giving $\Gamma_{i+1}[\![C_{i+1}]\!]$, and the assertion on σ is immediate. So suppose the rewrite proof to be non-trivial, and consider the first step in that proof. This step must take place in the cap part of $t\sigma$, because E is a constructor theory and $\Gamma_i[\![C_i]\!]$ obeys the constructor property; and this rewrite step can be lifted to a Cap Narrowing step since $\Gamma_i\tau$ is ground. □

5 Extension to Other Theories

It is natural to ask if the approach presented above can be extended to theories defined by rewrite systems R that are not convergent, dwindling, and constructor-based. Since (basic) R-narrowing is essential in our approach, it seems necessary to assume at least that R is convergent. On the other hand, the decidability of R-unification is necessary, in order that active deduction modulo R be decidable, cf. e.g. [9]. So any attempted extension should either be built over such an assumption, or should provide an implicit proof for the decidability of R-unification.

The Dolev-Yao system DY is the very basic intruder theory for which our preceding approach applies. Homomorphic Encryption (HE) is the theory that just extends DY by specifying that encryption distributes over pairs. HE can be

defined by the following system, which is no longer convergent, nor dwindling, nor a constructor system:

$$\begin{array}{ll} \pi_1(p(x,y)) = x & e(p(x,y),z) = p(e(x,z),e(y,z)) \quad (E_{homo}) \\ \pi_2(p(x,y)) = y & d(p(x,y),z) = p(d(x,z),d(y,z)) \quad (D_{homo}) \\ d(e(x,y),y) = x & \end{array}$$

The idea is to apply the inference rules exactly as before for the DY part of the theory, but to treat the $E_{homo} \cup D_{homo}$ part of the theory using goal directed inference rules, as is done in syntactic theories:

Homomorphic Deduction

$$\frac{\Gamma \sqcup \{S \sqcup \{e(s_1,t_1), e(s_2,t_2), \ldots, e(s_n,t_n)\} \rhd_{(SYM,HE)} e(s,t)\}[\![C]\!]}{(\Gamma \sqcup \{\{s_1,s_2,\ldots,s_n\} \rhd_{(\{p,\pi_1,\pi_2\},\emptyset)} s\}[\![C]\!])\sigma}$$

where σ is the most general unifier of $\{t_1 = t\} \cup \{t_2 = t\} \cup, \ldots, \cup \{t_n = t\}$ modulo the HE theory. None of our terms will contain a symbol π_1, π_2 or d, so this in effect means these equations only need to be solved modulo the equation E_{homo}, which itself forms a syntactic theory. Unification has been shown to be decidable for the convergent extension of HE containing an additional rule $e(d(x,y),y) = x$, in [2]; but unification modulo E_{homo} is easier. As in syntactic theories, we will just need an explicit Decomposition rule and an explicit Cap Decomposition rule in our inference system.

A further point on the Homomorphic Deduction rule is that the conclusion of the inference requires solving a Cap unification problem where the only symbols allowed to be used for caps are π_1, π_2 and p. The same inference rules as above (for the cap constraints under the full signature) can be used to solve these cap constraints. The main difficulty of such an inference system is that, after the Homomorphic Deduction rule, the No Information Loss property may not be true anymore. This means that equational constraints in normal form may not always have a solution. So some additional inference rules will be required to solve equational constraints in normal form. This is part of our on-going work; the details will appear elsewhere.

Note also that passive deduction is known to be decidable for (the convergent extension of) HE, as well as for *all* dwindling convergent systems, without exception, cf. [1]. Getting rid of the constructor assumption on dwindling systems that we have made in this paper, is an issue that is being looked into in [13].

6 Related Work, Conclusion

Several protocol decision procedures have been designed for handling equational properties [14, 8, 7] of the cryptographic primitives. Some works have tried to derive generic decidability results for some specific *class* of intruder theories. Delaune and Jacquemard [10] consider the class of *public collapsing* theories. These theories have to be presented by rewrite systems where the rhs of every rule is a ground term or a variable. Some results assume that the rhs of any rule is only a subterm of the lhs. [6] gives some results for related theories. The

procedure in [5] is more general, but more complex than ours; more importantly, it is not clear if it can be extended to the theory of homomorphic encryption.

Concerning theories with a homomorphism operator, beside a few results for passive intruders (e.g. [1]), the only work for active intruders is [11], which presents decidability results for a class of monoidal theories containing exclusive or, in combination with the homomorphism axiom. Their approach follows a classical schema for cryptographic protocol analysis, which proves first a locality result. The insecurity problem is then reduced to solving some linear Diophantine equations in a suitable algebra.

References

1. S. Anantharaman, P. Narendran, M. Rusinowitch. "Intruders with Caps". In *Proc. Int. Conf. RTA'07*, LNCS 4533, pp.20–35, Springer-Verlag, June 2007.
2. S. Anantharaman, H. Lin, C. Lynch, P. Narendran, M. Rusinowitch. "Equational and Cap Unification for Intrusion Analysis". LIFO-Research Report, `www.univ-orleans.fr/lifo/prodsci/rapports/RR/RR2008/RR-2008-03.pdf`
3. A. Armando, L. Compagna. SATMC: a SAT-based Model Checker for Security Protocols, In *Proc. of JELIA 2004*, LNCS 3229, pp. 730–733, Springer-Verlag, 2004.
4. D. Basin, S. Mödersheim, and L. Viganò. An On-The-Fly Model-Checker for Security Protocol Analysis. In Einar Snekkenes and Dieter Gollmann, editors, *Proceedings of ESORICS'03*, LNCS 2808, pages 253–270. Springer-Verlag, 2003.
5. M. Baudet. "Deciding security of protocols against off-line guessing attacks". In *Proc. of ACM Conf. on Computer and Communications Security*, 2005, pp. 16-25.
6. Y. Chevalier, M. Kourjieh. "Key Substitution in the Symbolic Analysis of Cryptographic Protocols". In *Proc. Int. Conf FSTTCS'07*, LNCS 4855, pp. 121–132, Springer-Verlag, December 2007.
7. Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani. "An NP Decision Procedure for Protocol Insecurity with XOR". In *Proceedings of the Logic In Computer Science Conference, LICS'03*, pages 261–270, 2003.
8. H. Comon-Lundh and V. Shmatikov. Intruder Deductions, Constraint Solving and Insecurity Decision in Presence of Exclusive or. In *Proceedings of the Logic In Computer Science Conference, LICS'03*, pages 271–280, 2003.
9. V. Cortier, S. Delaune, P. Lafourcade. "A Survey of Algebraic Properties Used in Cryptographic Protocols". In *Journal of Computer Security* 14(1): 1–43, 2006.
10. S. Delaune and F. Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS'04)*, pages 278–287, Washington, D.C., USA, October 2004. ACM Press.
11. S. Delaune, P. Lafourcade, D. Lugiez, and R. Treinen, Symbolic protocol analysis for monoidal equational theories, Information and Computation, 2008. To appear.
12. J.M. Hullot. Canonical Forms and Unification. CADE 1980: 318-334
13. H. Lin. PhD Thesis, Clarkson University, Potsdam, NY (USA), To appear (2008).
14. C. Meadows and P. Narendran. A unification algorithm for the group Diffie-Hellman protocol. In *Workshop on Issues in the Theory of Security (in conjunction with POPL'02), Portland, Oregon, USA, January 14-15*, 2002.
15. P. Narendran, F. Pfenning, R. Statman. On the unification problem for cartesian closed categories. In *Proc. of the Logic in Computer Science Conference LICS'93*, pages 57–63, 1993.

16. O. Pereira and J.J. Quisquater On the perfect encryption assumption. In: P. Degano, Editor, Workshop on Issues in the Theory of Security (WITS 2000), Geneva, Switzerland (2000), pp. 42-45.
17. C. Weidenbach. Towards an automatic analysis of security protocols. In Proc. of *16th International Conference on Automated Deduction, CADE-16*, LNAI 1632 (H. Ganzinger, ed.), Springer-Verlag, pages 378–382, 1999.

7 Appendix

The following convergent rewrite system defines the theory of Homomorphic Encryption (HE):

$$p_1(x\,.\,y) \to x \qquad\qquad enc(x\,.\,y, z) \to enc(x, z)\,.\,enc(y, z))$$
$$p_2(x\,.\,y) \to y \qquad\qquad dec(x\,.\,y, z) \to dec(x, z)\,.\,dec(y, z))$$
$$enc(dec(x, y), y) \to x$$
$$dec(enc(x, y), y) \to x$$

Theorem 4. *Unification modulo the theory HE is decidable.*

For the proof, we shall be applying several reductions on the given unification problem. To start with, we shall assume (via usual arguments, and reasoning mod HE) that the given unification problem $\mathcal{P}$ is in a *standard form*, in the following sense: each of its equations to solve, modulo HE, is assumed to have one of the following forms:

$$Z = T, \quad Z = X.Y, \quad Z = enc(X, Y), \quad Z = const.,$$

where the $T, X, Y, Z, \ldots$ stand for variables, and *const* is any free ground constant. (If an equation in $\mathcal{P}$ is given in the form $U = dec(V, W)$, it is rewritten mod HE as $V = enc(U, W)$.) The equations in $\mathcal{P}$ of the first (resp. second) form are said to be '*equalities*' (resp. '*pairings*'), those of the third form are said to be of the *enc* type, and the last ones of the 'constant' type. The second arguments of *enc*, in the equations of $\mathcal{P}$, are referred to as the *key variables* of $\mathcal{P}$.

The *conjugate* of any *enc* equation $Z = enc(X, Y)$ in $\mathcal{P}$, is defined as the equation $X = dec(Z, Y)$, said to be of the '*dec*' type. For every key variable/constant Y occurring in $\mathcal{P}$, let h_Y (resp. $\overline{h}_Y$) denote the homomorphism $enc(-, Y)$ (resp. $dec(-, Y)$) defined on terms. An *enc* equation $Z = enc(X, Y)$ can thus be written as $Z = h_Y(X)$, and its conjugate as $X = \overline{h}_Y(Z)$. Let n be the number of distinct key variables/constants appearing in $\mathcal{P}$, and let $\mathcal{H}$ stand for the set of all homomorphisms ($2n$ in number), thus associated with these key variables/constants.

We construct next a graph of dependency $G = G_{\mathcal{P}}$ between the variables of the problem $\mathcal{P}$. Its nodes will be the variables (or constants) of $\mathcal{P}$. From a node Z on G, there is an oriented arc to a node X iff the following holds:

a) $\mathcal{P}$ has an equation of the form $Z = h_i(X)$, or $Z = \overline{h}_i(X)$, for some $i \in \{1, \ldots, n\}$; the arc is then labeled with the symbol h_i (resp. with $\overline{h}_i$);
b) $\mathcal{P}$ has an equation of the form $Z = X.V$ (resp. $Z = V.X$): the arc is then labelled with p_1 (resp. with p_2).

Semantics: If G contains an edge of the form $Z \rightarrow^h X$, then Z can be evaluated by applying the homomorphism h to the evaluation of X.

With a view to eliminate redundancy on the graph, it is explicitly assumed that *variables which are 'equal' in $\mathcal{P}$ have exactly one representative node on G*; consequently, G will have no equality edges.

For solving $\mathcal{P}$, we shall make a few *assumptions* which express necessary conditions for the problem $\mathcal{P}$ to admit a *solution in normal form modulo HE*. The first among them is the following, that results from the so-called *Perfect Encryption* hypothesis:

(**SNF**): For any loop γ on G from some node Z to itself, the word formed by the symbols labeling the arcs composing γ must simplify to the empty word, under the following set of rules:

$$(\#) \qquad h_i\overline{h}_i \rightarrow \epsilon, \quad \overline{h}_i h_i \rightarrow \epsilon, \qquad 1 \leq i \leq n.$$

A problem $\mathcal{P}$ (in standard form) will be said to be *admissible* iff it satisfies SNF. The problems we consider will all be assumed to be admissible (a justification for this, based on algebraic considerations, can be found in Section 7.1). For such problems $\mathcal{P}$, it follows in particular from SNF, that:

· there can be no loop containing an arc labeled with a p_1 or p_2, from any node to itself, on the graph $G = G_{\mathcal{P}}$ ("the pairing operator is *free* in HE").

Several reductions, called *trimming*, will be applied to our problems $\mathcal{P}$. Some of them result from the Perfect Encryption assumption; but their prinicipal aim is to ensure that the non-key variables of $\mathcal{P}$ do not get split into pairs.

We define two relations on the set of variables $\mathcal{X} = \mathcal{X}(\mathcal{P})$ of $\mathcal{P}$:

- $Z \approx X$ iff one can go from Z to X by using only the equalities of $\mathcal{P}$, and the *enc* or *dec* arcs of $G_{\mathcal{P}}$;

- $Z \succ X$ iff there is a loop-free path from Z to X (on the graph of $\mathcal{P}$), with at least one of its arcs labeled with p_1 or p_2.

Rules for Trimming: We denote by **Eq** (resp. **Pair**, **Enc**) the set of equalities (resp. pairings, the *enc*-equations) in $\mathcal{P}$, respectively.

Rule 1. (*Perfect Encryption*)

$$a) \ \frac{\mathbf{Eq};\ \mathbf{Pair};\ \mathbf{Enc} \sqcup \{Z = enc(X,Y),\ Z = enc(V,Y)\}}{\mathbf{Eq} \cup \{V = X\};\ \mathbf{Pair};\ \mathbf{Enc} \sqcup \{Z = enc(X,Y)\}}$$

$$b) \ \frac{\mathbf{Eq};\ \mathbf{Pair};\ \mathbf{Enc} \sqcup \{Z = enc(X,Y),\ Z = enc(X,T)\}}{\mathbf{Eq} \cup \{T = Y\};\ \mathbf{Pair};\ \mathbf{Enc} \sqcup \{Z = enc(X,Y)\}}$$

Rule 1'. (*Variable Elimination*)

$$a) \ \frac{\{U = V\} \sqcup \mathbf{Eq};\ \mathbf{Pair};\ \mathbf{Enc}}{\{U = V\} \cup [V/U](\mathbf{Eq});\ [V/U](\mathbf{Pair});\ [V/U](\mathbf{Enc})}$$

Rule 2. (*Pairing is free in HE*)

$$a) \ \frac{\mathbf{Eq};\ \mathbf{Pair} \sqcup \{Z = U_1.U_2,\ Z = V_1.V_2\};\ \mathbf{Enc}}{\mathbf{Eq} \cup \{V_1 = U_1,\ V_2 = U_2\};\ \mathbf{Pair} \sqcup \{Z = U_1.U_2\};\ \mathbf{Enc}}$$

$$b)\ \frac{\mathbf{Eq};\ \mathbf{Pair};\ \mathbf{Enc};\quad Z \approx Z' \ \ and \ \ Z \succ Z'}{FAIL}$$

Rule 3. (*Split on Pairs*)

$$a)\ \frac{\mathbf{Eq};\ \mathbf{Pair};\ \mathbf{Enc} \sqcup \{Z = enc(X,Y)\};\quad Z = Z_1.Z_2 \in \mathbf{Pair}}{\mathbf{Eq};;\ \mathbf{Pair} \sqcup \{X = X_1.X_2\};\ \mathbf{Enc} \sqcup \{Z_1 = enc(X_1,Y),\ Z_2 = enc(X_2,Y)\}}$$

$$b)\ \frac{\mathbf{Eq};\ \mathbf{Pair};\ \mathbf{Enc} \sqcup \{Z = enc(X,Y)\};\quad X = X_1.X_2 \in \mathbf{Pair}}{\mathbf{Eq};\ \mathbf{Pair} \sqcup \{Z = Z_1.Z_2\};\ \mathbf{Enc} \sqcup \{Z_1 = enc(X_1,Y),\ Z_2 = enc(X_2,Y)\}}$$

(The X_1, X_2 in rule 3a, and the Z_1, Z_2 in rule 3b, are fresh variables – as indicated by the notation.) At any stage of the inference process, Rule $1'$ is meant to keep the graph of the 'current' problem irredundant, and Rule $2b$ checks for the SNF assumption. *The inference rules 1,1', 2 are all to be applied eagerly*; if rule $2b$ leads to 'FAIL', then the procedure stops. At any stage of the process, all the inferences under $1a, 1b$ and $2a$ are to be derived 'in block', i.e., in parallel.

We must show that such an inference procedure terminates on any admissible problem. For that purpose, we need to define certain notions.

(1): For any problem $\mathcal{P}$ that satisfies SNF, the relation $\succ$ defined above on its set of variables $\mathcal{X}$, is a well-defined, strict, partial order on $\mathcal{X}$. For any such problem $\mathcal{P}$, and for any given $Z \in \mathcal{X}$, the *sp-depth of* Z – denoted as $spd(Z)$ – is defined as *the maximum number of* p_1*- or* p_2*-labeled arcs along the loop-free paths from* Z *to all possible* $\mathcal{X} \in \mathcal{X}$.

(2): Let $\mathcal{P}$ be any given admissible problem. We introduce a binary (infix) operator '$\circ$' (representing pairs, but denoted differently, to avoid confusion); and define $\mathcal{T}_p(\mathcal{P}) = \mathcal{T}_p$ as the set of all terms formed over $\mathcal{X}$, the symbol '$\circ$', and the set of all homomorphisms h_T – where T runs over all the key variables of $\mathcal{P}$.

- Any pairing $X = X_1.X_2$ in $\mathcal{P}$, is seen as a rewrite rule: $X \to X_1 \circ X_2$;
- Any equation $Z = enc(X,T)$ in $\mathcal{P}$ gives rise to two rewrite rules:
  $Z \to^{h_T} X$, and $X \to^{\overline{h}_T} Z$.

Rules of the former type will be called pairing rules; those of the latter type will be respectively called h-rules or $\overline{h}$-rules, with key T, and with target X for the first among them, and Z for the second. We define $\mathcal{R}_\mathcal{P}$ to be the rewrite system formed of all such rules. By a *critical configuration* wrt $\mathcal{R}_\mathcal{P}$, we mean any given pair of distinct rewrite rules of $\mathcal{R}_\mathcal{P}$ such that:

- both rules have the same variable $X \in \mathcal{X}_\mathcal{P}$ to their left;
- if one of them is a h-rule (resp. $\overline{h}$-rule), then the other rule must be a pairing rule or a h-rule (resp. pairing rule or a $\overline{h}$-rule);
- if both are h-rules (or $\overline{h}$-rules), they have the same key or the same target.

The common lhs variable of a critical configuration is referred to as its *peak*.

(3): To any critical configuration wrt $\mathcal{R}_\mathcal{P}$ (for any given problem $\mathcal{P}$), with $X \in \mathcal{X}$ as the peak, we associate an integer called its *weight*, as follows:

Type i) the rules in the configuration are both pairing rules:
the weight here is defined as $spd(X)$, the *sp*-depth of the peak.

Type ii) one rule is a pairing rule, and the other is a h- or $\overline{h}$-rule:
the weight here is $spd(X)$.
Type iii) the rules in the configuration are both h-rules (resp. both $\overline{h}$-rules):
the weight is n_X, where n_X is the number of nodes
to which there is a loop-free, non-empty path from X,
formed only of *enc*-arcs (resp. formed only of *dec*-arcs).

Lemma 9. *Trimming terminates on (admissible) problems.*

Proof. To any given admissible problem $\mathcal{P}$, we associate its set $CC(\mathcal{P})$ of all the critical configurations over $\mathcal{R}_{\mathcal{P}}$; and define a measure $m(\mathcal{P})$, as the lexicographic combination of 3 components: $m_1 = m_1(\mathcal{P}), m_2 = m_2(\mathcal{P}), m_3 = m_2(\mathcal{P})$, where:
- m_1 is the number of distinct key variables appearing in $\mathcal{P}$;
- m_2 is the *multiset of weights* of configurations of Type i) and ii) in $CC(\mathcal{P})$;
- m_3 is the *multiset of weights* of the configurations of Type iii) in $CC(\mathcal{P})$.

Consider then any inference on $\mathcal{P}$, by a rule other than $2b$ (which – applied whenever applicable – would yield 'FAIL'). Inference rule $1b$ lowers m_1, and leaves m_2, m_3 unchanged. Inference rules $1a$ and $2a$ will leave m_1 unchanged, will decrease m_3, and will either either leave unchanged, or decrease, m_2 (in particular, because *such inferences are all to be derived 'in block', i.e., simultaneously.*) And the two inference rules $3a, 3b$ will also leave m_1 unchanged, will decrease m_2, and will either leave unchanged, or decrease, m_3. □

The problem $\mathcal{P}$ is said to be *trimmed* iff it is saturated under rules 1, 2 and 3. Such a problem $\mathcal{P}$ gets actually divided into two sub-problems which can be treated 'almost' separately (as we shall be seeing farther down): one containing only the pairings and equalities of $\mathcal{P}$, and the other containing only its *enc* equations; this latter sub-problem will be referred to as the simple kernel – or just kernel – of $\mathcal{P}$. *A problem $\mathcal{P}$ will be said to be* simple *iff it is its own kernel.*

Example 1. i) The following problem is not in standard form:
$Z = T,\ \ Z = enc(X, Y),\ \ X = dec(T, Y),\ \ X = U.V,\ \ Y = Y_1.Y_2,\ \ \ Y_2 = a;$
we first put it in standard form:
$Z = T,\ \ \ Z = enc(X, Y),\ \ T = enc(X, Y),\ \ X = U.V,\ \ Y = Y_1.Y_2,\ \ Y_2 = a.$
Under redundancy elimination, we first get:
$T = Z,\ \ Z = enc(X, Y),\ \ X = U.V,\ \ Y = Y_1.a,\ \ Y_2 = a;$
which has one critical configuration, namely: $Z \leftarrow^{\overline{h}_Y} X \rightarrow U.V$.

Only a splitting inference is applicable (on Z), and the final trimmed equivalent is the following problem:

$$Z = T,\ \ Z = Z_1.Z_2,\ \ X = U.V,\ \ Y = Y_1.a,\ \ Y_2 = a,$$
$$Z_1 = enc(U, Y),\ \ Z_2 = enc(V, Y).$$

ii) The following problem:

$$Z = enc(X, Y),\ \ \ Y = enc(Z, T),\ \ \ T = enc(Z, W),\ \ \ Y = Y_1.Y_2.$$

is in standard form, but not trimmed: we have one critical configuration, namely: $Z \leftarrow^{h_T} Y \rightarrow Y_1.Y_2$, with peak at Y. Now $spd(Y) = 1$, but $n_Y = 3$ (we can go from Y to T, X, Z using only *enc*/*dec* arcs); so m_2 here is $\{3\}$, and the measure $m(\mathcal{P})$ of the problem is $(3, \{3\}, -)$.

Trimming needs here several splitting steps. We first write $Z = Z_1.Z_2$, and replace the second *enc* equation by the 2 equations: $Y_1 = enc(Z_1, T), Y_2 = enc(Z_2, T)$; we get a problem with two critical configurations, both with peak at Z, $spd(Z) = 1$ and $n_Z = 2$; so the measure is lowered to $(3, \{2, 2\}, -)$. Next, we write $X = X_1.X_2$ and replace the first *enc* equation by: $Z_1 = enc(X_1, Y), Z_2 = enc(X_2, Y)$, and get a problem with measure $(3, \{1\}, -)$. Finally, we write $T = T_1.T_2$ and replace the last *enc* equation by: $T_1 = enc(Z_1, W), T_2 = enc(Z_2, W)$. We thus get the following trimmed equivalent, with measure $(3, -, -)$:

$$\begin{array}{c} Z_1 = enc(X_1, Y), \quad Z_2 = enc(X_2, Y), \\ Y_1 = enc(Z_1, T), \quad Y_2 = enc(Z_2, T), \\ T_1 = enc(Z_1, W), \quad T_2 = enc(Z_2, W), \\ Y = Y_1.Y_2, , \quad Z = Z_1.Z_2, \quad X = X_1.X_2, \quad T = T_1.T_2. \end{array}$$

Remark 1. i) It is not hard to see, that if m is the number of equations in $\mathcal{P}$, and n the sum of the lengths of oriented loop-free paths on the dependency graph of $\mathcal{P}$, then trimming $\mathcal{P}$ terminates in at most $2\,mn$ steps. However, the number of equations in the trimmed equivalent, derived at the end, can be exponential wrt m; a typical illustrative example is the following:

$$\begin{array}{lll} X_1 = enc(X_2, U1) & X_{11} = enc(X_{12}, U2) & X_{111} = enc(X_{112}, U3) \\ X_1 = X_{11}.X_{12} & X_{11} = X_{111}.X_{112} & X_{111} = X_{1111}.X_{1112} \end{array}$$

ii) In view of the lemma above (and our non-redundancy assumption on the dependency graph), a trimmed problem is essentially an admissible problem $\mathcal{P}$ (with no critical configurations on pairings alone), such that:

- There is no node Z on the dependency graph of $\mathcal{P}$ from which there is an *outgoing* '*enc*' or '*dec*' arc, as well as an *outgoing* arc labeled with a p_1 *or* p_2.
- If X, V are any two distinct nodes on the graph of $\mathcal{P}$, then the equality $X = V$ is not an equality in $\mathcal{P}$.

iii) As a consequence, solving a trimmed problem $\mathcal{P}$ essentially reduces to solving its kernel $\mathcal{P}'$: any variable to the left of a pairing gets its solution by substituting from the solutions for the variables of the kernel.

iv) Observe that the key variables in any admissible problem $\mathcal{P}$, in standard form, remain as they are under splitting (as the Y, T, W in Example 1.ii) above); so their number remains unaffected by trimming. □

7.1 Solving a Simple Problem:

We henceforth assume all our problems $\mathcal{P}$ to be trimmed; and $\mathcal{P}'$ will stand for the kernel of $\mathcal{P}$. Our objective now is to conceive an algorithm for solving $\mathcal{P}'$. Note that the labels on the arcs of the sub-graph G' (of $G_{\mathcal{P}}$) of dependency for the problem $\mathcal{P}'$, are all in $\mathcal{H}$. Note also that, thanks to the SNF assumption on $\mathcal{P}$, there is a uniquely determined, loop-free, oriented path from any given node on G', to any other node on G'. An oriented path on G' will be said to be *maximal*, iff it cannot be extended to the left or to the right, i.e., it is not a proper sub-path of another oriented path of greater length. For solving $\mathcal{P}'$, we shall be using the maximal paths γ (on the connected components of G'), based (but only partly) on the following idea:

- first choose an end-node V on the path γ, and assign some value v to V;

- then, to every other node X along γ, assign the value derived by '*propagating that value from* V to X'; i.e., assign to X the value $\alpha(v)$ where α is the word over $\mathcal{H}$ that labels the oriented path from X to the chosen end-node V.

Such an idea has to be used, however, with some restrictive assumptions on γ. A first assumption we make is that a variable X of $\mathcal{P}'$ may *not* be 'evaluated' by using a word over $\mathcal{H}$ already containing h_X or $\overline{h}_X$; this corresponds to the 'occur-check' assumption in usual unification, over the empty theory.

Definition 11. *i) A maximal (oriented) loop-free path γ, on the dependency graph of G', is said to satisfy the condition* Occur-Check-Path *– or is said to* pass the test OCP *– if and only if the following holds:*
(**OCP**): *For any arc $U \rightarrow^h V$ on G' composing γ, with label $h = h_Y$ or $\overline{h}_Y$,* none of the nodes traversed by γ prior to U, is Y or a factor of Y for pairing.

However, checking for OCP along all the maximal paths – or their reversed paths – does *not* suffice give a complete procedure for solving simple problems, as is illustrated by several of the examples given below. We need a much more complex approach.

Let $\mathcal{P}$ be any 'simple' problem, with no pairing equations. We know that solving $\mathcal{P}$ amounts to solving the unification problem modulo the convergent system R formed of the following two rules:

$$enc(dec(x,y),y) \rightarrow x, \qquad dec(enc(x,y),y) \rightarrow x.$$

We may assume, wlog, that the key-variables of $\mathcal{P}$ are *all distinct* (more precisely: are assumed 'unequal') modulo R. We also assume, as we may, that the graph of $\mathcal{P}$ is connected.

Let $a, \bar{a}$ be rational numbers such that: $a\bar{a} = 1$, with $a \neq -1$. We set then: $b = -a$, $\bar{b} = 1$, and model the algebra of terms modulo R as the algebra $\mathcal{M} = \mathcal{M}_{\mathcal{P}}$ of linear polynomial expressions over the variables of $\mathcal{P}$, with coefficients in the ring $\Lambda = \mathbb{Z}[a, \bar{a}]$, where the homomorphisms via *enc* and *dec* are interpreted respectively as the morphisms:

$$h_V(x) = a\,x + b\,V, \qquad \overline{h}_V(x) = \bar{a}\,x + \bar{b}\,V.$$

From the assumption that the key-variables of $\mathcal{P}$ are different mod R, we then get the following consequences:

• Any given solution $\mathcal{P}$ (if there is one) can be seen as a substitution in the algebra $\mathcal{M}$, that solves for the non-key variables as polynomial expressions over the key-variables, with coefficients in the ring Λ.

• The key-variables of $\mathcal{P}$ are linearly *independent* over Λ, when seen as elements of the algebra $\mathcal{M}$.

Based on the latter fact, we derive an algebraic formulation for a criterion called *occur-check-graph* – referred to as OCGr below – for the problem $\mathcal{P}$ to be solvable. We start with the following observation: Let γ be any (loop-free) path on the graph $G = G_{\mathcal{P}}$, from a node X to some node Y on G, and let α be the word over the alphabet $\mathcal{H}$, labeling the arcs along the path γ; if the occur-check condition OCP holds along γ, then neither α nor any of its sub-words (seen

as morphisms on the algebra $\mathcal{M}$) can induce a non-trivial linear dependency between the key-variables of $\mathcal{P}$. Here is an illustrative example:

Example 2a. Consider the following problems:

$$Z = enc(X, X), \quad Z = dec(T, V)$$

Indeed, OCP does not fail along the (maximal) path *from* T *to* X; the first of the two equations solves for the non-key variable Z in terms of the key X, as $Z = h_X(X)$, and the second then solves for the non-key variable T as $T = h_V(Z) = h_V(h_X(X))$. To this 'corresponds' the fact that on the polynomial algebra $\mathcal{M}$, we get the following solutions for the non-keys Z, T, without inducing any dependency on the keys X, V:

$Z = \bar{a}\, X + \bar{b}\, X = \bar{a}\, X + X$, and

$T = aaX + abX + bV = aaX - aaX - aV = -aV$. □

It is important to note now, in addition, that if the word α, labeling a path from a node X to some node Y on $G = G_{\mathcal{P}}$, is such that neither α nor any of its sub-words induces non-trivial linear dependencies between the keys on the algebraic model $\mathcal{M}_{\mathcal{P}}$, then the same is true also for the reverse word α^{-1}, that labels the reverse path from Y to X on the graph G.

This leads us to the following algebraic formulation of the criterion OCGr, for a simple problem $\mathcal{P}$ to be solvable, under the assumption that *its keys are to be unequal* modulo the rewrite system R:

• (**OCGr**): let γ be any maximal loop-free path on $G = G_{\mathcal{P}}$ and let X and Y be its end-nodes; write $X = \alpha(Y)$, α being the word over $\mathcal{H}$ labeling (the arcs along) γ from X to Y; then neither α, nor any of its sub-words, may induce a non-trivial dependency between the key-variables of $\mathcal{P}$, as a morphism on $\mathcal{M}_{\mathcal{P}}$.

From our considerations above, it follows that OCGr is a necessary condition for a simple problem $\mathcal{P}$ to admit a solution with all its key-variables unequal modulo R.

Example 2b. Consider the following problems:

(i) $\mathcal{P}_1$: $\quad Y = enc(Z, X), \quad X = enc(Z, Y)$

(ii) $\mathcal{P}_2$: $\quad Z = enc(X, X), \quad Z = dec(T, T)$

(iii) $\mathcal{P}_3$: $\quad U = enc(X, Z), \quad Z = enc(U, Y), \quad Y = enc(U, X)$

(i) $\mathcal{P}_1$ is unsolvable: otherwise we would have a solution, say satisfying $Y = h_X\overline{h}_Y(X)$, which gives – when seen as a morphism over the algebra $\mathcal{M}$:

$$Y = a\bar{a}X + a\bar{b}Y + bX = X + aY - aX$$

which would be a non-trivial linear dependency on the keys X and Y that we have assumed to be unequal.

(ii) Problem $\mathcal{P}_2$ is unsolvable too: There is a path from X to T on the graph of the problem, labeled with word $\alpha = \overline{h}_X\overline{h}_T$. (Both T and X are keys, only Z is not a key.) Under the morphism over the algebra $\mathcal{M}$, we derive:

$$T = \overline{h}_X\overline{h}_T(X) = \bar{a}\bar{a}X + \bar{a}\bar{b}T + \bar{b}X = \bar{a}\bar{a}X + \bar{a}T + X,$$

which gives a linear dependency between T and X.

(iii) $\mathcal{P}_3$ is again unsolvable: each one of the 3 (maximal) paths between the key-variable nodes X, Y, Z is labeled by a word over $\mathcal{H}$, that would lead to a non-trivial dependency between these keys.

We assume henceforth that our simple problems $\mathcal{P}$ satisfy the criterion OCGr. Under the already made assumption that the keys of $\mathcal{P}$ are to be unequal modulo R, and the graph of $\mathcal{P}$ is connected, we then propose a method for solving $\mathcal{P}$.

We begin by defining a relation between the variables of such a $\mathcal{P}$ – denoted as $\succ_k$, and called *key dependency* – as follows:

• $Y \succ_k X$ iff $Y \neq X$ and the (unique, loop-free) path from Y to X on the graph $G_{\mathcal{P}}$ contains an arc labeled with h_X or $\overline{h}_X$. We then have the following:

Lemma 10. *If $\mathcal{P}$ is simple* (with keys all unequal) *satisfying OCGr, and if $G_{\mathcal{P}}$ is connected, then there can be* no key dependency cycle *on G: i.e., there cannot exist two distinct nodes X, Y on G such that $Y \succ_k X$ as well as $X \succ_k Y$.*

Proof. If G contains a key dependency cycle, say between two distinct nodes X, Y – which are then both keys as well as nodes–, then one can check (without difficulty) that the the word labeling the (unique, loop-free) path from X to Y would create a dependency involving the two keys X and Y, as a morphism on the algebra $\mathcal{M}_{\mathcal{P}}$; that would contradict the assumption OCGr. □

Definition 12. *Let $\mathcal{P}$ be simple* (with keys all unequal) *and satisfying OCGr, γ any maximal loop-free path on $G_{\mathcal{P}}$ and X', X'' its end-nodes. A node Y on γ is said to be a* base-node *for γ iff the occur-check condition OCP is satisfied along γ,* from *X' as well as* from *X'',* towards *the node Y.*

Lemma 11. *If $\mathcal{P}$ is simple* (with keys all unequal) *and satisfies OCGr, then every maximal path γ on the graph of $\mathcal{P}$ admits at least one base-node.*

Proof. If *every* node Y on γ fails to be a base-node for γ, then it is not hard to show that there is a sub-path of γ between two nodes X and Y, such that $Y \succ_k X$ as well as $X \succ_k Y$, which contradicts the previous lemma. □

Example 3. Consider the following problem:

$$(\mathcal{P})\text{: } X = enc(U, V), \quad U = enc(V, T), \quad V = enc(Y, U)$$

where U, T, V are the keys, and X, Y are not. The graph is connected with one maximal (loop-free) path γ going between the end-nodes X and Y; and it does not satisfy the OCP condition in either direction. However, γ does satisfy the OCGr condition: neither the word $h_V h_T h_U$ nor any of its sub-words, leads to a dependency relation between the keys. This problem $\mathcal{P}$ is actually solvable: suffices, for instance, to set:

$V = \overline{h}_T(U),\ Y = \overline{h}_U(V) = \overline{h}_U \overline{h}_T(U)$, taking U (and T) to be arbitrary,
and subsequently solve for X as: $X = h_V(U)$.

This solution has been obtained by taking U as a base-node for γ.

Suppose now, that we replace the key T by the key Y; then, the problem thus modified would be unsolvable: indeed, in that case, the sub-path going from U to Y would be labeled by the word $h_Y h_U$; so a likely solution would have to satisfy $U = h_Y h_U(U)$; on the algebra $\mathcal{M}_{\mathcal{P}}$ we would then get: $U = aaU + abU + bY =$

$aaY - aaU - aY = -aY$, giving a dependency between the keys U and Y; so γ in this case would fail to satisfy OCGr. □

Remark 2. A maximal path γ can (obviously) have more than one base-node. But if Y', Y'' are two base-nodes for γ, then they can be seen as 'equivalent' in the following sense:

- the occur-check condition OCP is satisfied along γ, from Y' towards Y'', as well as from Y'' towards Y';
- the keys of the arcs on the sub-path of γ between Y' and Y'' are not nodes outside this sub-path.

It follows then, that any intermediary node on the sub-path between Y' and Y'' is also a base-node for γ.

We extend now the notion of base-node, to that of a *cbase-node* (meaning "covering base-node") for a set of nodes:

Definition 13. *Let $\mathcal{P}$ be simple* (with keys all unequal) *satisfying OCGr, Y any node on the graph $G = G_{\mathcal{P}}$, and S a given set of nodes on G (on the same connected component as Y). Then Y is said to be a* cbase-node *for the set S, iff:*

(i) Y is a base-node for some maximal path on G; and

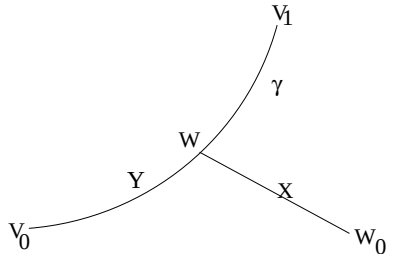
(ii) for any $X \in S$, Y is also a base-node for the maximal path on G passing through X and Y.

For instance, for the simple problem: $Z = enc(X, Y),\ Y = enc(Z, T),\ T = enc(Z, W)$ studied in detail in Example 4 below, the node T is a cbase-node for the set of all nodes on the graph of the problem.

Now, given a problem $\mathcal{P}$ as above, let $\mathcal{S} = \mathcal{S}_{\mathcal{P}}$ denote the set of all nodes Y on its graph, such that Y is a cbase-node for some non-empty set S_Y of nodes on G. The set $\mathcal{S}$ is non-empty: it contains all the base-nodes for the maximal paths on G. We define then a partial order $\preceq$ on the set $\mathcal{S}$ by setting: $U \preceq V$ iff $S_U \subseteq S_V$. We have then the following:

Lemma 12. *Let $\mathcal{P}$ be simple* (with keys all unequal) *satisfying OCGr, and let Y be any node in the set $\mathcal{S} = \mathcal{S}_{\mathcal{P}}$, on some connected component Γ of $G_{\mathcal{P}}$, such that S_Y is $\preceq$-maximal in $\mathcal{S}$; then $S_Y = \Gamma$.*

Proof. Let Y be $\preceq$-maximal in $\mathcal{S}$, and assume that there is a node $X \in \Gamma$ that is not in S_Y. Let γ be a maximal path on Γ for which Y is a base-node. Since Γ is connected, there is a unique path of minimal length that joins X to some node W on γ; let W_0 be the other end-node of a maximal path extending the (unique) path from V_0 to X (or from V_1 to X), in the notation of the figure below:



Since Y is a base-node for γ, the occur-check-path condition OCP is satisfied from V_0 and from V_1 towards Y, along γ. On the other hand, the maximal path from V_0 to W_0 also admits a base-node W' (by Lemma 11), and the only case to consider would be when this base-node W' is not on γ itself, i.e., lies somewhere between W and W_0, and is not W. This in particular implies that OCP is satisfied from Y towards W, so we may assume wlog that $Y = W$.

Now, if OCP is also satisfied from W' to $W = Y$, then $S_Y = S_W$ would contain the node W', so would also contain the node X, contrary to what we assumed. So, OCP must fail from W' towards W. But W' is a base-node for the path from V_0 to W_0, so along this path OCP must hold from V_0 towards W', which means that $S_{W'}$ contains in particular $W = Y$, so contains strictly S_Y; but this would contradict the $\preceq$-maximality of the cbase-node Y. □

Definition 14. *Let Γ be any connected component on the graph G of a simple problem $\mathcal{P}$, and $V_0 \in \mathcal{S}_{\mathcal{P}}$ such that $S_{V_0} = \Gamma$. Then V_0 is called a* base-variable *for $\mathcal{P}$ on the connected component Γ.*

Corollary 2. *If $\mathcal{P}$ is simple,* (with keys all unequal) *and satisfies OCGr, then, $\mathcal{P}$ is solvable.*

Proof. On every given connected component Γ of the graph $G = G_{\mathcal{P}}$ of $\mathcal{P}$, we choose some base-variable V; by definition of base-variable, for any given node X on Γ, the unique path on G from X to V must satisfy the condition OCP; we solve for X by propagating to X any value v that is assignable to the chosen base-variable V: i.e., we set $X = \alpha_{XV}(v)$, where α_{XV} is the word over $\mathcal{H}$ labeling the path from X to V. □

We are in a position, now, to formulate a non-deterministic decision procedure for solving any HE-unification problem, given in standard form.

7.2 Solving a Problem in Standard Form: The Algorithm $\mathcal{A}$

Given: $\mathcal{P}$ = a HE-unification problem $\mathcal{P}$, given in standard form.
G = the dependency graph for $\mathcal{P}$.

1a. If G does *not* satisfy SNF, or contains two equations of the form $Z = a, Z = b$, where a, b are two different constants, exit with 'Fail'.
1b. Guess (non-deterministically) a set of equal keys, and assume all the other key-variables of $\mathcal{P}$ to be 'unequal' mod HE.
1c. Apply the Trimming Inferences to $\mathcal{P}$; if this leads to FAIL, exit with 'Fail';
1d. Else replace $\mathcal{P}$ by a trimmed equivalent;
$\mathcal{P}'$ = the kernel of $\mathcal{P}$; G' = the sub-graph of G for $\mathcal{P}'$.
2a. Check for the criterion OCGr on every connected component of G';
2b. If OCGr is unsatisfied on some component, exit with 'Fail';
3a. For each connected component Γ of G', *guess a base-variable* V_γ.
3b. Build a substitution for the variables on each component Γ: assign to V_γ some term, then to every other node $X \in \Gamma$ the value derived by propagation from V_γ to X. Let σ' be the substitution, solution for $\mathcal{P}'$, thus obtained.

4. Propagate the values deduced from σ' to the variables (of the equalities and pairings) of G, that are not in G';
   - if inconsistency, exit with 'Fail';
   - else return σ = substitution thus obtained, as solution to $\mathcal{P}$.

The algorithm $\mathcal{A}$ is of cost NP with respect to the number of equations of the simple kernel $\mathcal{P}'$ of the problem $\mathcal{P}$; this is so, because the failure of the OCGr criterion on some path can be guessed in time NP wrt the number nodes on G'. The soundness of the algorithm $\mathcal{A}$ follows from the fact that each of its steps is syntactically coherent with $\mathcal{P}$; and its completeness results from the fact that the OCGr criterion is a condition necessary for any simple problem to be solvable (when the distinct key variables of $\mathcal{P}$ are all assumed unequal modulo HE).

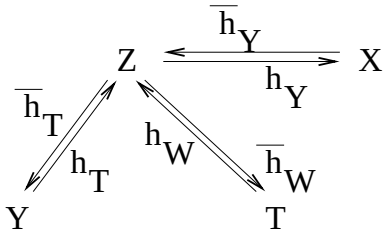
7.3 Some Illustrative Examples

Note first that the substitution that the algorithm $\mathcal{A}$ returns as a solution for a problem $\mathcal{P}$, is built "in a lazy style" in its steps $3a$ through 4: the variables are left uninstantiated, in general; they get instantiated only if/when needed (cf. e.g., Example 4 below).

Example 4. Consider the following problem:

$(\mathcal{P}')$: $Z = enc(X, Y), \quad Y = enc(Z, T), \quad T = enc(Z, W).$

The problem is simple, and its dependency graph is connected:



Node Y cannot be a base-variable, since the path from T to Y contains an arc labeled with $\overline{h}_T$; similarly, X cannot be a base-variable. We have node T as the only base-variable here. We solve for Z and Y, along the path from Y to T (that satisfies OCP, by definition): namely $Y \rightarrow^{h_T} Z \rightarrow^{\overline{h}_W} T$; choosing arbitrarily T, W we get $Z = \overline{h}_W(T), Y = h_T\overline{h}_W(T)$ as solutions for Z, Y; and for the variable X, connected to this path at Z, we deduce get $X = \overline{h}_Y(Z) = \overline{h}_Y\overline{h}_W(T)$. (Note: the base-variable T has not been assigned any specific term here.)

Suppose now, the problem $(\mathcal{P}')$ is the kernel of a non-simple problem, e.g.:

$(\mathcal{P})$: $Z = enc(X, Y), \quad Y = enc(Z, T), \quad T = enc(Z, W). \quad X = a$

Then, for the above solution for its simple kernel to be valid, we need to check if $a = \overline{h}_Y\overline{h}_W(T)$ holds; this can be done by instantiating T now, as $h_Y h_W(a)$. □

Example 5. The following two problems are simple:

i) $X = enc(Y, X), \quad Y = enc(Z, Z)$

ii) $X = enc(Y, T)$, $Y = enc(Z, X)$, $Z = enc(W, V)$, $W = enc(V, S)$
and their graphs are connected.

Problem i) is unsolvable: one cannot have $dec(X, X) = enc(Z, Z)$ modulo HE, whatever be X, Z. But this unsolvability can also be checked by observing that the condition OCGr cannot be satisfied.

Problem ii) is unsolvable too; and it is not hard to check that it will fail to satisfy OCGr, no matter which keys are 'made equal'. □