Cap Unification: Application to Protocol Security modulo Homomorphic Encryption *

Siva Anantharaman
LIFO, Université d'Orléans
Orléans, France
siva@univ-orleans.fr

Hai Lin
Clarkson University
Potsdam, NY, USA
linh@clarkson.edu

Christopher Lynch
Clarkson University
Potsdam, NY, USA
clynch@clarkson.edu

Paliath Narendran
University at Albany-SUNY
Albany, NY, USA
dran@cs.albany.edu

Michael Rusinowitch
Loria-INRIA Grand Est
Nancy, France
rusi@loria.fr

ABSTRACT

We address the insecurity problem for cryptographic protocols, for an active intruder and a bounded number of sessions. The protocol steps are modeled as rigid Horn clauses, and the intruder abilities as an equational theory. The problem of active intrusion – such as whether a secret term can be derived, possibly via interaction with the honest participants of the protocol – is then formulated as a Cap Unification problem. Cap Unification is an extension of Equational Unification: look for a cap to be placed on a given set of terms, so as to unify it with a given term modulo the equational theory. We give a decision procedure for Cap Unification, when the intruder capabilities are modeled as homomorphic encryption theory. Our procedure can be employed in a simple manner to detect attacks exploiting some properties of block ciphers.

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., [20, 3, 13]). Reducing the security problem to a constraint solving problem 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 convergent rewrite systems that are collapsing (right-hand sides are variables), expressing simply that decryption cancels encryption when provided with the right key. Then, extensions of narrowing techniques for semantic unification [15] can be applied to solve the constraints derived from the cryptographic protocol and the secrecy property that one wants to check. Several protocol decision procedures have been designed for handling more equational properties of the cryptographic primitives [17, 8, 7] Some works have tried to derive generic decidability results for some specific *class* of intruder theories [11, 5, 6]. These results address theories presented by rewrite systems where the rhs (right hand side) of every rule is a ground term, a variable or a subterm of the lhs. Concerning theories with a homomorphism operator, the only work for *active* intruders is [12], 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 (see, e.g. [8]). The insecurity problem is then reduced to solving some linear Diophantine equations in a suitable algebra. It must be noted that none of these approaches handle homomorphic encryption over the pair operator.

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.

The paper is structured as follows: Section 2 presents the preliminary notions, and in particular the basic Dolev-Yao rewrite system DY, and a system HE for homomorphic encryption (non-convergent, as it is). The notions of *Cap Constraints* and *Cap Unification* are introduced in Section 3. In Section 4, we present an inference procedure to *decide* unification modulo a (homomorphic) theory induced by a convergent, single rule, subsystem $\mathcal{E}_h$ of HE; the procedure

*Work Supported by NSF Grants CNS-0831305, CNS-0831209, and by AVANTSSAR FP7-ICT-2007-1 Project 216471

ASIACCS'10 April 13–16, 2010, Beijing, China.

is essentially syntactic, and is simpler than that given in [19] for Unification modulo One-sided Distributivity. The inference system for active deduction modulo HE is given in Section 5; the idea is to reduce the problem of solving cap constraints over HE, via narrowing, to one of solving cap constraints over the single rule theory $\mathcal{E}_h$, and eventually to solving an $\mathcal{E}_h$-unification problem. The technique employed is simpler than those used in [5, 6], and we come up with a procedure for solving cap constraints which is shown to be sound, terminating and complete for our homomorphic encryption theory HE. The cap constraints modeling the protocols are assumed to satisfy some minor restrictive assumptions satisfied by all usual protocols.

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*. We assume the signature to have finitely many constants. A rewrite rule is a pair of terms (l, r) such that $l \succ r$, for some given reduction ordering [1] $\succ$ 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.

In this paper, we are concerned with the insecurity problem of protocols, for instance, the problem where a message intended as secret is captured or deduced by an intruder. We model the homomorphic encryption theory as a convergent rewrite system R, that is a *constructor system*. By that we mean: the signature Σ is a disjoint union $\Sigma_D \sqcup \Sigma_C$, the symbols in Σ_C are called *constructors*, those in Σ_D are called *defined symbols;* the top symbols of all left hand sides (lhs) of the rules of R are defined symbols, 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. An inference system, called saturation of the cap closure, was given in [1] for passive deduction, and was shown to be complete for Dolev-Yao theories and a convergent theory of Homomorphic Encryption ("Encryption distributes over pairs"). Our concern in this paper is Dolev-Yao plus Homomorphic Encryption in the case of active intruders. The following Dolev-Yao theory DY, with signature $\Sigma = \{\pi_1, \pi_2, p, e, d\}$, underlies all known formalisms for passive or active deduction:

[1] A reduction ordering is a well-founded ordering, stable under contexts and substitutions.

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

The homomorphic encryption theory that we consider – denoted as HE in the sequel – extends DY with the following rule:

$$e(p(x, y), z) \to p(e(x, z), e(y, z))$$

In these theories, '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. It is important to note that our HE differs from the system considered in [1] and in [2]: the following two rules

$$e(d(x, y), y) \to x,\ d(p(x, y), z) \to p(d(x, z), d(y, z))$$

are *not included* in ours. And as it is, our HE is *not* convergent; but, as we shall be seeing farther down, it suffices to add one 'meta'-reduction rule to get a convergent system that is equivalent. In our approach developed below, the convergent subsystem of HE consisting of the single rule $e(p(x, y), z) \to p(e(x, z), e(y, z))$ will be playing a crucial role; it will be denoted as $\mathcal{E}_h$. Intruder knowledge evolution is modeled as 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 $f(t_1, t_2, \ldots, t_n) \in Cap(S, SYM)$.*

(It is assumed in the definition above, that if f is π_1 or π_2, then its argument t must be a pair.) 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 t are all terms; it will be denoted as $\{t_1, \ldots, t_n\} \blacktriangleright t$.*

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 $\mathcal{E}$ the associated equational theory, a protocol rule $\{t_1, \ldots, t_n\} \blacktriangleright t$ is said to be an *R- or $\mathcal{E}$-constructed* protocol rule if no function symbol in the rule is a defined symbol of $\mathcal{E}$.

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 that of a single session, via standard techniques ([11]).

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 often said to be 'rigid' variables.

Our next step will be to model every step of a protocol session as a Cap Constraint, and propose a technique called Cap Unification, to solve the set of all such constraints. (Note: Cap constraints have also been called "Deducibility constraints" in many related works; cf. e.g. [8, 18].)

3. CAP CONSTRAINTS

In this section, R is any given, convergent, rewrite system over some signature Σ, $\mathcal{E}$ the equational theory of R, and SYM is any given set of symbols from Σ.

DEFINITION 3. *A* cap constraint *is a constraint written in the form $S \triangleright_{(SYM,\mathcal{E})} t$, where S is a set of terms, and t is a term. It is* solvable *iff there exists $s \in Cap(S, SYM)$, and a substitution σ s.t. $s\sigma = t\sigma \bmod \mathcal{E}$. We call σ a solution of $S \triangleright_{(SYM,\mathcal{E})} t$.*

An $\mathcal{E}$-equation (or just 'equation') is, as usual, an $\mathcal{E}$-equality constraint of the form $s =_{\mathcal{E}} t$, where s and t are terms; if the theory $\mathcal{E}$ is obvious from the context, we simply write $s = t$; for ease and uniformity of presentation, we agree to identify it with the 'special' cap constraint $s \triangleright_{(SYM,\mathcal{E})} t$, 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 $\mathcal{E}$-unifying s and t.

DEFINITION 4. *Let $\Gamma = \{S_i \triangleright_{(SYM,\mathcal{E})} t_i,\ 1 \le i \le n\}$, be any set of cap constraints (some of which may be special). A substitution σ is a* solution *for Γ iff σ is a solution for* every *cap constraint in Γ.*

From Protocol to Cap Constraints:.

We show here how to generate a set of cap constraints from a strand space that 'describes' a protocol session. The idea is similar to that given in [18]. We begin with a definition, essentially as in [14]:

DEFINITION 5. *(i) Let A be a set, the elements of which are the possible messages that can be exchanged between principals in some given protocol session P. A signed term is a pair $\langle \sigma, a \rangle$ with $a \in A$, and $\sigma \in \{+, -\}$. As usual, $(\pm A)^*$ is the set of all finite sequences of signed terms, an element of which will be typically denoted as $\langle\langle \sigma_1, a_1 \rangle, \ldots, \langle \sigma_n, a_n \rangle\rangle$. A signed term $\langle \sigma, t \rangle$ is generally written as $+t$ or $-t$.*

(ii) A strand space for the protocol session P is then defined as a finite set $\mathcal{S}$ of signed terms (over some given signature), together with a trace mapping $tr : \mathcal{S} \to (\pm A)^*$.

Given a strand space $\mathcal{S}$ for P, we define a node n as a pair $\langle s, i \rangle$, with $s \in \mathcal{S}$ and i an integer satisfying $1 \le i \le length(tr(s))$; for any such node n, we set $index(n) = i$, and $sterm(n) = s$. The set of all nodes will be denoted by N. If $n_1, n_2 \in N$, then $n_1 \to n_2$ means $sterm(n_1) = +a$ and $sterm(n_2) = -a$ for some $a \in A$. Semantically this is to be seen as: node n_1 sends the message a, which is received by n_2, thus creating a causal link between their strands. If $n_1, n_2 \in N$, then $n_1 \Rightarrow n_2$ means n_1, n_2 occur on the same strand with $index(n_1) = index(n_2) - 1$. In other words, n_1 is an immediate causal predecessor of n_2 on the strand.

A strand space describing a protocol session P can be seen as a directed graph G with two types of edges: $n_1 \to n_2$ and $n_1 \Rightarrow n_2$. For our purposes here, we shall treat both types of edges alike. It should be clear that the graph is acyclic and has a unique linear ordering (corresponding to the sequential ordering of the messages exchanged, during the given protocol session). We denote by $Pos(G)$ (resp. $Neg(G)$) the total number of positive (resp. negative) nodes in G; and by $pos_i(G)$ (resp. $neg_i(G)$) the i-th positive (resp. negative) node on G under the linear ordering. In intuitive terms: $pos_i(G)$ (resp. $neg_i(G)$) corresponds to the message sent (resp. received) at the i-th step of the protocol session.

Let $\mathcal{S}$ be a strand space for a protocol session P, and let G be the corresponding directed graph. Then, to any given deduction problem on P, and any given set of random natural numbers $rand = \{i_1, i_2, \ldots, i_n \mid i_k < Pos(G), 1 \le k \le n\}$, we can associate, in a natural manner, a set of cap constraints. (Intuitively $rand$ is a nondeterministic guess on which message exchanges are useful to the intruder; the idea is similar to that given in [18].) For instance, let the random sequence $rand$ be given, and suppose the deduction problem is whether an intruder gets to know a message m intended secret for him/her. Let K_{init} be the set of (ground) terms forming the intruder's initial knowledge. We then generate the following cap constraints (where, by Σ we mean a superset of the set of symbols in the intruder theory $\mathcal{E}$):

$$\lambda_k = K_{init} \cup \{term(pos_i(G)) \mid i \in rand,\ i < i_k\} \triangleright_{(\Sigma,\mathcal{E})} term(neg_k(G)),$$
for every $1 \le k \le n$; and
$$\lambda_{n+1} = K_{init} \cup \{term(pos_i(G)) \mid i \in rand\} \triangleright_{(\Sigma,\mathcal{E})} m$$

where $term(pos_i(G))$ (resp. $term(neg_k(G))$) denotes the unsigned (usual) term at the i-th (resp. k-th) node on the graph G.

Illustrative Example: As a concrete example, we consider the following "NEEDHAM-SCHROEDER SYMMETRIC KEY PROTOCOL" ([9]), which aims to establish a fresh shared symmetric key K_{ab}, for mutual authentication of the participants: in any session, the value of K_{ab} is to be known only to the participants playing the roles of A, B and S in that session.

A, B, S: principals
N_a, N_b: fresh nonces
K_{as}, K_{bs}: symmetric keys
succ: number $\to$ number
1. $A \to S : A, B, N_a$
2. $S \to A : \{N_a, B, K_{ab}, \{K_{ab}, A\}K_{bs}\}_{K_{as}}$
3. $A \to B : \{K_{ab}, A\}_{K_{bs}}$
4. $B \to A : \{N_b\}_{K_{ab}}$
5. $A \to B : \{succ(N_b)\}_{K_{ab}}$

The strand trace for A is:

$\langle +p(p(A,B),N_a), -T, +y, -\{N_b\}_z, \{succ(N_b)\}_z\rangle$,
where $T = e(p(p(N_a,B), p(K_{ab}, e(p(k_{ab},A),K_{bs}))), K_{as})$.
The strand trace for B is: $\langle -y, +\{N_b\}_z, -\{succ(N_b)\}_z\rangle$.
The strand trace for S is: $\langle +p(p(A,B),N_a), +T\rangle$.

When encryption is based on the ECB block chaining technique (i.e., performed sequentially on a block decomposition of the plaintext, and under the assumption that messages are assigned a round number of blocks), it can be seen as an homomorphism on 'pair'.

The following is an attack based on homomorphism, where A can be fooled into accepting the publicly known nonce N_a as a secret key shared with B.

i.1. $A \rightarrow S : A, B, N_a$
i.2. $S \rightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$
ii.3. $I(B) \rightarrow A : \{N_a, B\}_{K_{as}}$
ii.4. $A \rightarrow I(B) : \{N'_a\}_{N_a}$
ii.5. $I(B) \rightarrow A : \{succ(N'_a)\}_{N_a}$

The point of the above attack – a priori on authentication, but susceptible to lead to one on secrecy – is that we can answer "yes" to the following two questions.

i) Can the intruder know the concatenation of A, B and N_a based on the initial knowledge $\{A, B, N_a\}$?

ii) Can the intruder get to know both a message of the form $e(p(x,B),K_{as})$ and x itself? (If yes, then the intruder can convince A to accept x – that (s)he already knows – as a secret key to be shared with B.)

As discussed above, we set $p(e(p(x,B),K_{as}),x)$ as the secret message (the goal). If we set $rand = \{1\}$, we then generate the following constraints, corresponding to the two questions above.

i. $\{A, B, N_a\} \rhd_{(\{p,\pi_1,\pi_2,e,d\},HE)} p(p(A,B),N_a)$
ii. $\{A, B, N_a, e(p(p(p(N_a,B),K_{ab}), e(p(K_{ab},A),K_{bs})), K_{as})\}$
$\rhd_{(\{p,\pi_1,\pi_2,e,d\},HE)} p(e(p(x,B),K_{as}),x)$

In Section 5.3, we shall show our cap unification procedure can be applied to solve these cap constraints.

4. UNIFICATION MODULO $\mathcal{E}_H$

Our ultimate objective is an algorithm for solving cap constraints over HE, which in particular will also solve unification problems modulo HE. For that, we first need an algorithm for unification modulo the theory $\mathcal{E}_h$ defined by the single rule $e(p(x,y),z) \rightarrow p(e(x,z),e(y,z))$; we give here an inference procedure that is simpler than that given in [19] for Unification modulo One-sided Distributivity. Referred to as $\mathcal{E}_h$-*Unifn* in the sequel, it is a generalization of the standard algorithm for syntactic unification. To the rest of this section, Γ represents a set of equations modulo $\mathcal{E}_h$; and $=$ will denote equality modulo $\mathcal{E}_h$. (Syntactic equality will be denoted as $=_{id}$, when necessary.)

We first formulate some 'standard' syntactic inference rules dealing with usual unification. Here we don't consider '=' as oriented; in other words, $x = t$ and $t = x$ are considered the same.

- (**Trivial**) $\Gamma \sqcup \{t = t\} \Rightarrow \Gamma$
- (**Std Decomposition**)
$\Gamma \sqcup \{f(s_1, s_2, \ldots, s_m) = f(t_1, t_2, \ldots, t_m)\} \Rightarrow$
$\Gamma \cup \{s_1 = t_1\} \cup \{s_2 = t_2\} \cup \cdots \cup \{s_m = t_m\}$
- (**Variable Substitution**) $\Gamma \sqcup \{x = t\} \Rightarrow \Gamma\sigma \cup \{x = t\}$
if $x \notin Vars(t)$, x occurs in Γ, and $\sigma = [x \mapsto t]$.
- (**Clash**) $\Gamma \sqcup \{f(s_1, \ldots, s_m) = g(t_1, \ldots, t_n)\} \Rightarrow fail$
if $f \neq g$; and if one of f and g is 'e'(resp. 'p'), then the other is not 'p'(resp. 'e').
- (**Occur Check**) $\Gamma \sqcup \{x = t\} \Rightarrow fail$,
if $t \neq x$ and $x \in Vars(t)$.

Note: The Std Decomposition rule covers, in particular, the so-called Perfect Encryption assumption:
$e(x,k) = e(y,k) \Rightarrow x = y$; $\quad e(x,k) = e(x,k') \Rightarrow k = k'$.

Over the empty theory, two terms with different function symbols on top do not unify; but, modulo $\mathcal{E}_h$, a term with p on top may unify with a term with e on top. To handle this, we shall introduce below some additional rules, referred to as *Homomorphic Pattern* rules. We shall be needing a few new notions, and some notational convention, for their formulation.

Preliminaries for Homomorphic Pattern rules:.
We begin with the following definition.

DEFINITION 6. *The positions of a single variable term x is $pos(x) = \{\epsilon\}$; and the positions of a term $f(t_1, \cdots, t_n)$ are: $pos(f(t_1, \cdots, t_n)) = \{\epsilon\} \cup \{i \cdot p | p \in pos(t_i), 1 \leq i \leq n\}$*

So a position is a sequence of integers indicating a path in the tree representation of a term. For each position q in a term t, we define new functions `ppos` and `epos` which represent the subsequence of q representing p symbols and e symbols respectively.

We define $\texttt{ppos}(q,t)$ inductively as follows. The base case is $\texttt{ppos}(\epsilon,t) = \epsilon$. The inductive step is $\texttt{ppos}(i \cdot q, t) = i \cdot \texttt{ppos}(q, t|_i)$ if the symbol at the top of t is 'p', and $\texttt{ppos}(i \cdot q, t) = \texttt{ppos}(q, t|_i)$ if the symbol at the top of t is 'e'.

Similarly we define $\texttt{epos}(q,t)$ such that $\texttt{epos}(\epsilon,t) = \epsilon$; and, $\texttt{epos}(i \cdot q, t) = i \cdot \texttt{epos}(q, t|_i)$ if the symbol at the top of t is 'e', and $\texttt{epos}(i \cdot q, t) = \texttt{epos}(q, t|_i)$ if the top symbol of t is 'p'.

For example, consider the term $t = p(e(p(x,y),k),z)$; the variable y is at position 112 on t. We have:
$\texttt{ppos}(112,t) = 1.\texttt{ppos}(12, e(p(x,y),k))$
$= 1.\texttt{ppos}(2, p(x,y)) = 1.2$;
$\texttt{epos}(112,t) = \texttt{epos}(12, e(p(x,y),k)$
$= 1.\texttt{epos}(2, p(x,y)) = 1$.

Let t_1, t_2 be any two terms, q_1 a position in t_1, and q_2 a position in t_2. The position-terms (q_1,t_1) and (q_2,t_2) are said to be *incompatible* if at least one of the following holds:

1. $\texttt{ppos}(q_1,t_1)$ is a proper prefix of $\texttt{ppos}(q_2,t_2)$; *or*
2. $\texttt{ppos}(q_1,t_1) = \texttt{ppos}(q_2,t_2)$, $\texttt{epos}(q_1,t_1) \neq \texttt{epos}(q_2,t_2)$ and $\texttt{epos}(q_1,t_1)$ and $\texttt{epos}(q_2,t_2)$ contain only 1's in their sequences.

Two position-term pairs are said to be *compatible* if they are not incompatible.

A pair of terms t_1, t_2 are said to be *in phase* iff for all positions q_1 in t_1 and q_2 in t_2 such that $t_1|_{q_1} = t_2|_{q_2}$, the position-term pairs $(q_1, t_1), (q_2, t_2)$ are compatible. The terms t_1 and t_2 are said to be *out of phase* iff they are not in phase; they are *out of phase on variable* x iff there exist positions q_1 in t_1 and q_2 in t_2 such that $t_1|_{q_1} = t_2|_{q_2} = x$ and the position-term pairs $(q_1, t_1), (q_2, t_2)$ are incompatible. (Note: This actually is a *generalized occur-check condition* on the variable x.) Note that every term is in phase with itself. A position q in a term t is said to be a *non-key* position iff $\texttt{epos}(q, t)$ contains only 1's. For any two terms that are equivalent to each other modulo $\mathcal{E}_h$, we have the following result:

LEMMA 1. *Let s and t be terms such that $\mathcal{E}_h \models s = t$. Then s and t are in phase on variables at non-key positions.*

PROOF. Let s and t be terms such that $\mathcal{E}_h \models s = t$, then there is an equational proof of $s = t$ using the equation $\mathcal{E}_h$. Suppose that this proof has n steps. We will prove that s and t are in phase, by induction on n. In the base case, suppose that $n = 0$, then s is syntactically equal to t, so s and t are in phase. For the induction step, note that $\mathcal{E}_h$ preserves the property of being in phase. □

If two terms s, t are out of phase on some variable, then for any substitution σ, the terms $s\sigma, t\sigma$ are out of phase. The Homomorphic Pattern inferences will contain a 'Failure' rule, such that the unification problem will 'Fail' if applied to an equation $s = t$ where s and t are out of phase on a variable at a non-key position. (Note however, that this is an "if", and not an "iff": indeed any two different constants a, b are in phase.)

Next we show, that the non-key positions of t_1 and t_2 have a certain relationship if t_1 is equivalent to t_2 modulo $\mathcal{E}_h$.

LEMMA 2. *Let t_1 and t_2 be terms such that $\mathcal{E}_h \models t_1 = t_2$. Let q_1 be a non-key position in t_1. Then there exists a position q_2 in t_2 such that $\texttt{ppos}(q_1, t_1) = \texttt{ppos}(q_2, t_2)$ and $\texttt{epos}(q_1, t_1) = \texttt{epos}(q_2, t_2)$.*

PROOF. Since $\mathcal{E}_h \models t_1 = t_2$, there is an equational proof of $t_1 = t_2$ using the single equation $\mathcal{E}_h$. Suppose that this proof has n steps. We prove by induction on n that the above properties hold. If $n = 0$, then t_1 is syntactically equal to t_2, so we can set $q_2 = q_1$. For the induction step, suffices to note that $\mathcal{E}_h$ preserves the properties to be proved. □

The Homomorphic Pattern rules will also incorporate the following principle: if $e(x, k)$ is a pair, then the argument encrypted – namely x – must itself split as a pair. But with a view to *not* increase the number of unsolved variables of the problem, these rules will be formulated as macro (or hyper) rules, which group several such inferences into one single inference.

The macro rules will be formulated by using a suitable representation of terms, using two *new symbols*: E and P_v, where v is some sequence of finite strings over the alphabet $\{1, 2\}$; we agree to refer to such sequences simply as *bit string sequences*. Informally, a term with P_v on top is a certain representation for a term with p as top-symbol; similarly, a term with E on top represents a term with e as top-symbol. For instance, the P-representation of the 'usual' term $p(p(e(a, k), e(b, k')), e(c, k''))$ is:

$$P_{11,12,2}(e(a, k), e(b, k'), e(c, k'')).$$

And the E-representation of the usual term $e(e(a, k_1), k_2)$ is $E(a, k_1, k_2)$. But only 'legal' bit string sequences v can lead to meaningful terms with p on top. Such sequences are defined, inductively, as follows[2]:

· The empty string ϵ is a legal bit string sequence (it is of length 0).

· If $v = a_1, ..., a_n$ and $w = b_1, ..., b_m$ are legal sequences of respective lengths n, m, then $1.v, 2.w = 1a_1, ..., 1a_n, 2b_1, ..., 2b_m$ is a legal sequence, of length $n + m$.

We now define formally the P- and E-representations of a term:

(P): Define a position q in a term t to be a *pure p-position* in t iff $epos(q, t) = \epsilon$. We say that q is a *maximal pure p-position* in t iff q is a pure p-position in t, and q is not a proper prefix of any pure p position in t.

For instance, let $t = p(p(a, b), e(p(c, d), k))$, then the pure p-positions in t are the positions where the subterms t, $p(a, b)$, a, b and $e(p(c, d), k)$ occur. And the maximal pure p-positions in t are the positions of the subterms a, b and $e(p(c, d), k)$. The *P-representation* of a term t is then $P_{q_1,\cdots,q_n}(t_1, \cdots, t_n)$ where $\{q_1, \cdots, q_n\}$ is the lexicographically ordered set of maximal pure p-positions in t, and for all i, $t|_{q_i} = t_i$.

(E): Define a position q in term t to be a *pure e-position* in t if $ppos(q, t) = \epsilon$. If q is a pure e-position in t, and q either contains no 2 at all or contains 2 only as the last element of the sequence, then q is said to be a *penuk-position* in t. (*penuk* abbreviates '*pure e-position not under a key*'.) We say that q is a *maximal penuk-position in t* if q is a *penuk*-position in t, and q is not a proper prefix of any *penuk*-position in t.

For instance, if $t = e(e(a, b), e(c, d))$, then every position in t is a pure e-position in t. Every position in t is a *penuk*-position except for 21 and 22 where c and d occur. The maximal *penuk* positions are the positions where a, b and $e(c, d)$ occur. (The reader can now see why the name *penuk*: the terms c and d are inside the key $e(c, d)$ that encrypts the message $e(a, b)$.)

The *E-representation* of a term t is then $E(t_1, \cdots, t_n)$ where $\{q_1, \cdots, q_n\}$ is the set of all maximal *penuk*-positions in t written in lexicographic order, and for all i, $t|_{q_i} = t_i$.

Remark: The P-representation (resp. E-representation) of a usual term t normalizes to its usual representation, under the rewrite rules **P** (resp. **E**) below :

(**P**): $P_\epsilon(t) \to t$

$P_{1.v,2.w}(t_1, \ldots, t_{n+m}) \to p(P_v(t_1, \ldots, t_n), P_w(t_{n+1}, \ldots, t_{n+m}))$

if v (resp. w) is a legal bit string sequence of length n (resp. m).

[2]These bit strings correspond to leaf positions on binary trees

(**E**): $E(t) \to t$

$E(t, k_1, \ldots, k_{n-1}, k_n) \to e(E(t, k_1, \ldots, k_{n-1}), k_n)$

We may now formulate the Homomorphic Pattern rules.

- (**Shaping**)

  $\Gamma \sqcup \{P_v(t_1, \ldots, E(x, k_m, \ldots, k_n), \ldots, t_l) = E(s, k'_1, \ldots, k'_n)\}$
  $\Rightarrow$
  $\Gamma \cup \{P_v(t_1, \ldots, E(x', k'_1, \ldots, k'_{m-1}, k_m, \ldots, k_n), \ldots, t_l) = E(s, k'_1, \ldots, k'_n)\} \cup \{x = E(x', k'_1, \ldots, k'_{m-1})\}$

  where x' is a fresh variable, v is a (legal) bit string sequence, and $n \geq m > 1$.

- (**Failure**)

  (i) $\Gamma \sqcup \{s = t\} \Rightarrow Fail$

  if s, t are out of phase on some variable at a non-key position (i.e., with `epos` containing only 1's). [3]

  (ii) $\Gamma \sqcup \{P_v(t_1, \ldots, E(t_i, k_1, \ldots, k_m), \ldots, t_l) = E(s, k'_1, \ldots, k'_n)\} \Rightarrow Fail$

  where v is a (legal) bit string sequence, t_i not a variable, and $m < n$.

- (**Parsing**)

  $\Gamma \sqcup \{P_v(E(t_1, k_{11}, \ldots, k_{1m_1}), \ldots, E(t_l, k_{l1}, \ldots, k_{lm_l})) = E(s, k_1, \ldots, k_m)\}$
  $\Rightarrow$
  $\Gamma \sqcup \{P_v(E(t_1, k_{11}, \ldots, k_{1m_1-1}), \ldots, E(t_l, k_{l1}, \ldots, k_{lm_l-1})) = E(s, k_1, \ldots, k_{m-1})\} \cup \{k_{1m_1} = k_{2m_2} = \cdots = k_{lm_l} = k_m\}$

  where v is a (legal) bit string sequence.

The Homomorphic Pattern rules *are to be performed en bloc* together with the *Variable Substitution* rule given above, by which we mean: whenever one of these 'Homomorphic Pattern' rules or the 'Variable Substitution' rule applies, none among the remaining rules shall be applied. The 'Shaping' rule helps make the terms being unified to be 'well-structured'. The 'Parsing' rule takes a pair of 'well-structured' terms, and solves for them with a macro inference, based on the above mentioned principle of 'encryption distributes over pairs'. The 'Failure' rule tries to detect failure as early as possible, and *is always applied the most eagerly among all rules.* Failure rule (i) is sound by Lemma 1, while Failure rule (ii) corresponds to the case where the two terms considered have different numbers of keys, but Shaping is inapplicable.

A simple example of Failure (i) rule is for $p(x, y) = e(x, k)$. The x in $p(x, y)$ is considered as an encryption with zero keys. The x in $e(x, k)$ is in an encryption with 1 key. By the footnote of Failure (i) we fail because $0 < 1$. Note also that $p(x, y)$ and $e(x, k)$ are out of phase on x.

The $\mathcal{E}_h$-*Unifn* procedure is defined by all the above rules and the rules for syntactic unification. We explicitly *make the following assumption:* the equations of our problems are given in $\mathcal{E}_h$-normal form, and any new equation derived under the inferences is kept in $\mathcal{E}_h$-normal form.

A *solved form* for $\mathcal{E}_h$-Unifn is a set of $\mathcal{E}_h$-equalities $\{x_1 = t_1, \cdots, x_n = t_n\}$, where each x_i is a *solved variable,* each t_i is a usual term. (Note: a variable x, in a set of equality constraints, 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.) A solved form for $s = t$ is denoted as $nf(s = t)$. We shall be showing below that two terms s and t are unifiable modulo $\mathcal{E}_h$ if and only if there exists a solved form for $s = t$.

For any given set of $\mathcal{E}_h$-unification problems Γ, let $uvars(\Gamma)$ be the number of distinct *unsolved* variables in Γ. For any equality S appearing in Γ, let $Terms(S)$ be the multiset $\{s \mid s \text{ appears in } S\}$; and let $Unifns(Terms(\Gamma))$ be the multiset $\{Terms(S) \mid S \in \Gamma\}$. We then define a measure $M(\Gamma)$ for the cap unification problem Γ as the lexicographic pair $(uvars(\Gamma), Unifns(Terms(\Gamma)))$.

LEMMA 3. *Let Γ be any set of $\mathcal{E}_h$-unification problems. All the above inference rules for Unification modulo $\mathcal{E}_h$ reduce $M(\Gamma)$.*

PROOF. We show that the Homomorphic Pattern rules reduce $M(\Gamma)$; it is straightforward that all the other rules reduce the measure.

No rule in Homomorphic Pattern increases the number of unsolved variables. We show that, if any of the above rules increases $Unifns(Terms(\Gamma))$, then it must decrease $uvars(\Gamma)$.

Among the inference rules, Shaping is the only one that can increase $Unifns(Terms(\Gamma))$, since a variable x gets then replaced by $E(x', k_1, \ldots, k_m)$, where x' a new variable.

To show this, we first observe that the Shaping rule cannot be applied if $s = x$, where s and x are as in the Shaping Rule: indeed, in that case, let q_1 be the position of x in the entire term on the LHS, say T_1, and q_2 the position of s in the term on the RHS, say T_2. Then $\mathtt{ppos}(q_2, T_2) = \epsilon$, so $\mathtt{ppos}(q_2, T_2)$ is a prefix of $\mathtt{ppos}(q_1, T_1)$. Also, $\mathtt{epos}(q_1, T_1) < \mathtt{epos}(q_2, T_2)$. Therefore, T_1 and T_2 are out of phase on x, so we would have failed.

Therefore, we will assume that $s \neq x$ when the Shaping rule is applied. This means that the Shaping rule can be applied only finitely many times before some Parsing rule is triggered. This is because any application of the Shaping rule gives rise to at least one of the $t'_i s$ having at least n encryption keys (n as in the formulation of the Shaping rule), so the Parsing rule will eventually become applicable.

Consider then the application of the Parsing Rule, where s is now as in the formulation of the Parsing Rule. By definition of E, we require that the top symbol of s is not e. Since the terms are in normal form, the top symbol of s cannot be p. Therefore s must be a variable or a constant. If s is a variable, then Variable Substitution is immediately applied, reducing the number of unsolved variables. If s is a constant c, then suppose that Shaping has added a new variable x'. So the result of shaping will either be of the form $p(r_1, r_2) = c$, in which case we will fail, or it will be of

[3] As we will see in the Completeness proof, we only need to check if s is of the form $P_v(\ldots, E(x, k_{i1}, \ldots, k_{im_i}), \ldots)$, t is of the form $E(x, k_1, \ldots, k_m)$ and $m_i < m$. That is the only 'out of phase' condition that is necessary for completeness.

the form $x' = c$, in which case Variable Substitution will be applied, and the number of *unsolved variables* will decrease. (Note that Shaping will have to be applied *en bloc* with Parsing, by assumption.) □

The next two lemmas show that $\mathcal{E}_h$-Unifn is sound and complete. As usual, we write $\Gamma \Rightarrow \Gamma'$ if the problem Γ' is derived from the problem Γ by applying one of the inferences.

LEMMA 4. *Let σ be any 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.*

PROOF. We show the soundness of Homomorphic Pattern rules, the soundness of all the other rules being straightforward. Specifically we show that the Parsing rule of Homomorphic Pattern is sound.

Now, according to the homomorphic encryption theory, $p(e(t_1, k_1), e(t_2, k_2)) = e(s, k)$ if and only if $s = p(t_1, t_2)$, and $k_1 = k$, $k_2 = k$. This can be generalized without difficulty to $P_v(E(t_1, k_{11}, \ldots, k_{1m}), \ldots, E(t_n, k_{n1}, \ldots, k_{nm})) = E(s, k_1, \ldots, k_m)$ if and only if $s = P_v(t_1, \ldots, t_n), k_{11} = \cdots = k_{n1} = k_1, \ldots, k_{1m} = \cdots = k_{nm} = k_m$, where v is any legal bit string sequence. □

LEMMA 5. *Let Γ_1 be a set of unification problems that is not in solved form, and σ a 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 can be extended to a solution of Γ_2.*

PROOF. Consider any unsolved unification problem c in Γ_1, of the form $s = t$. If s or t is a variable, Variable Substitution applies. If neither of them is a variable and they have the same function symbols on top, then Std Decomposition applies.

So we assume they have different function symbols on top; then Homomorphic Pattern rules apply. Suppose then the equation $s = t$ is of the form:

$$P_v(E(t_1, k_{11}, \ldots, k_{1m_1}), \ldots, E(t_l, k_{l1}, \ldots, k_{lm_l})) = E(s', k_1, \ldots, k_m).$$

Case 1): Suppose there is an $i, 1 \leq i \leq l$, with $m_i < m$.

Case 1.a) Case where t_i is a variable:

Case 1.a.i): Suppose $t_i = s'$. Then s and t are out of phase on the variable t_i. So, the Failure Rule (i) applies.

Case 1.a.ii): Suppose $t_i \neq s'$. Then Shaping applies. Clearly, any solution to Γ_2 can be extended to a solution of Γ_1.

Case 1.b) Case where t_i is not a variable: Then Failure Rule (ii) applies. The term t_i cannot be a p-term, because s is in normal form modulo $\mathcal{E}_h$. So t_i must be a constant. Let q_2 be the position of s' in t. No instance of s can have a term at position q_2 by Lemma 2, thus the equation $s = t$ has no solution.

Case 2): For all i, $m_i \geq m$. In this case, Parsing applies, and the solution to Γ_2 is a solution of Γ_1, because the equation $\mathcal{E}_h$ implies that $E(P_v(x_1, \ldots, x_n), z_1, \ldots, z_m) = P_v(E(x_1, z_1, \ldots, z_m), \cdots, E(x_n, z_1, \ldots, z_m))$. □

LEMMA 6. *Let Γ be a set of unification problems, σ any substitution solution for Γ modulo $\mathcal{E}_h$. Then there is a solved form σ' for Γ, produced by $\mathcal{E}_h$-Unifn, that generalizes σ, and σ' only contains variables in Γ.*

PROOF. By Lemma 5, the inference steps of $\mathcal{E}_h$-Unifn preserve solutions. Lemma 3 shows that $\mathcal{E}_h$-Unifn terminates.

Only the Shaping rule introduces new variables. By the argumentation of Lemma 3, the Shaping rule always triggers the Parsing rule, which gets rid of the introduced variables. □

We close this section with an example, which shows why we need to keep the equations of our problems in $\mathcal{E}_h$-normal form:

Example: Consider the following $\mathcal{E}_h$-unification problem:

$$e(p(x, y), k) = e(e(w, k), k).$$

Its term to the right (with 2 keys) is in normal form; and the term to the left, with one key, is not in normal form. Shaping appears inapplicable, but Failure rule (ii) is. However, the equation in $\mathcal{E}_h$-normal form actually reads:

$$p(e(x, k), e(y, k)) = e(e(w, k), k),$$

to which neither of the Failure rules applies, but Shaping does apply. After some Shaping inferences, followed by Variable Substitution, the problem becomes:

$x = e(x', k), y = e(y', k),$
$p(e(e(x', k), k), e(e(y', k), k)) = e(e(w, k), k).$

Parsing then solves the problem as:

$w = p(x', y'), x = e(x', k), y = e(y', k)$ □

5. ACTIVE DEDUCTION MODULO HE

We turn our attention now to active deduction modulo our theory HE, and present an inference system to solve a set of cap constraints modulo HE. The idea is to formulate its rules as 'calling' the $\mathcal{E}_h$-Unifn rule of the previous section. Such an inference system will be shown to be sound and complete for HE.

For the rest of the section, *SYM* will stand for either $sig(HE)$ or $\{p, \pi_1, \pi_2\}$. The inference system for active deduction modulo HE, denoted as $\mathcal{I}_D$, consists of the rules given below, where Γ stands for any set of cap constraints. The inferences will be applied *starting with an initial set of* cap constraints *modeling the protocol clauses*, so containing no equality constraints. Equality constraints that may have to be considered during the inferences will all be over $\mathcal{E}_h$, so '=' will stand for equality modulo $\mathcal{E}_h$. A *SYM* that appears in the premise and the conclusion of any rule, stands for the *same* given symbol set.

Projection:

$$\frac{\Gamma \sqcup \{S \sqcup p(t_1, t_2)\} \rhd_{(SYM,HE)} t\}}{\Gamma \cup \{S \cup \{t_1, t_2\} \rhd_{(SYM,HE)} t\}}$$

Decryption:

$$\frac{\Gamma \sqcup \{S \sqcup \{e(t, k)\} \rhd_{(SYM,HE)} t'\}}{\Gamma \cup \{S \cup \{t\} \rhd_{(SYM,HE)} t'\} \cup \{S \rhd_{(SYM,HE)} k\}}$$

if $d \in SYM$.

Degeneracy:

$$\frac{\Gamma \sqcup \{S \rhd_{(SYM,HE)} t\}}{\Gamma\tau}$$

if $S = \{s_1, \cdots, s_n\}$, and $\tau = nf(s_i = t) \neq \bot$ for *some* i in $\{1, \ldots, n\}$.

Homomorphic Deduction:

$$\frac{\Gamma \sqcup \{S \sqcup \{e(s_1,t_1), e(s_2,t_2), \ldots, e(s_n,t_n)\} \rhd_{(SYM,HE)} e(s,t)\}}{\Gamma \cup \{S \cup \{s_1, s_2, \ldots, s_n\} \rhd_{(\{p,\pi_1,\pi_2\},HE)} s\}\tau}$$

where τ is some $nf(\{t_1 = t\} \cup \{t_2 = t\} \cup \cdots \cup \{t_n = t\})$.

Cap Decomposition:

$$\frac{\Gamma \sqcup \{S \rhd_{(SYM,HE)} f(t_1, \ldots, t_m)\}}{\Gamma \cup \{S \rhd_{(SYM,HE)} t_1\} \cup \cdots \cup \{S \rhd_{(SYM,HE)} t_m\}}$$

if $f \in SYM$.

Variable Substitution:

$$\frac{\Gamma}{\Gamma\sigma}$$

where (i) $x \in Vars(\Gamma)$, $\sigma : x \mapsto P_v(t_1, \ldots, t_n)$ for a legal bit string sequence v of length n;
(ii) and the t_i's are *distinct, non-variable* terms in the LHS of the constraints in Γ, such that $x \notin Vars(t_i), 1 \leq i \leq n$.

The *Degeneracy rule* corresponds to the case where one of the terms to the left of a cap constraint in Γ is $\mathcal{E}_h$-unifiable with the term to the right of that constraint; *this rule is to be applied eagerly.* The inference rules of $\mathcal{I}_D$ are *don't-know nondeterministic*: i.e., for completeness, they may all have to be tried, in turn. As usual, we write an inference as a transformation $\Gamma \Rightarrow_{\mathcal{I}_D} \Gamma'$ on sets of cap constraints. We shall show below that $\mathcal{I}_D$-derivations – which are, by definition, sequences of such transformations – terminate and are sound; and also that $\mathcal{I}_D$ *is complete for satisfiability*; i.e., if Γ is solvable, then there is an $\mathcal{I}_D$-derivation from Γ to an *empty* set of cap constraints Γ'.

5.1 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 all usual protocols; they will be shown to be preserved under the inferences $\mathcal{I}_D$.

DEFINITION 7. *A set of* cap constraints $\Gamma = \{S_1 \rhd_{(SYM,HE)} t_1, \ldots, S_m \rhd_{(SYM,HE)} t_m\}$ *is said to satisfy the* Standard Protocol Property*, iff the following two conditions are satisfied:*

1. *The* Variable Introduction *property: Let* $1 \leq j \leq m$*, and* x *any variable that appears in* S_j*; then there exists* $i, 1 \leq i \leq n, i \neq j$*, such that* $t_i = x$*.*
2. *The* Constructor *property: None of the symbols* d, π_1, π_2 *appears in* Γ*.*

We shall assume that the initial set of cap constraints modeling the protocol rules have the Standard Protocol Property: Variable introduction means that a principal's actions are determined by the messages (s)he receives or deduces. The constructor property says that the protocol clauses do not contain functions that destruct data.

Our purpose in this subsection is to show the termination of any $\mathcal{I}_D$-derivation. For that, we first need to show that $\mathcal{I}_D$-derivation sequences preserve the Standard Protocol Property.

LEMMA 7. *Let* $\Gamma_0, \Gamma_1, \ldots, \Gamma_n$ *be an* $\mathcal{I}_D$*-derivation, where* Γ_0 *has the Variable Introduction property. Then* Γ_n *also has the Variable Introduction property.*

PROOF. We will show that if $\Gamma \Rightarrow_{\mathcal{I}_D} \Gamma'$, and Γ has the Variable Introduction property, then so does Γ'. By inspection of all the inference rules, if some occurrence of a variable disappears, then that variable has to be instantiated. We consider two cases:

Case 1: If no variable in Γ is instantiated, then all occurrences of all variables in Γ still remain in Γ'. The statement is trivially true.

Case 2: If some variable is instantiated, that variable will be instantiated everywhere, according to our inference rules. So there will be no occurrence of that variable in Γ'. The statement is also true in this case. □

LEMMA 8. *Suppose* Γ *has the Constructor property, and* $\Gamma \Rightarrow_{\mathcal{I}_D} \Gamma'$*. Then* Γ' *also has the Constructor property.*

PROOF. By inspection of the inference rules, any symbol appearing in the conclusion of an inference has to appear already in its premise. □

Proving Termination:.

For any set Γ of cap constraints, we define: $nvars(\Gamma) = |\{x \mid x \text{ appears in } \Gamma\}|$. For any cap constraint S appearing in Γ, let $Terms(S)$ be the multiset $\{s \mid s \text{ appears } S\}$, and $Constraints(\Gamma)$ be the multiset $\{Terms(S) \mid S \in \Gamma\}$. We then define the measure of Γ as the lexicographically ordered pair: $M(\Gamma) = (nvars(\Gamma), Constraints(\Gamma))$. This measure is well-founded. We show now that it is reduced by every $\mathcal{I}_D$-inference.

LEMMA 9. *If* Γ *has the Standard Protocol property, and if* $\Gamma \Rightarrow_{\mathcal{I}_D} \Gamma'$*, then* $M(\Gamma') < M(\Gamma)$*.*

PROOF. Projection and Decryption do not increase the first component, but reduces the second component. By Lemma 6, Homomorphic Deduction, Degeneracy and Cap Decomposition do not increase the first component. Actually, Homomorphic Deduction either reduces the first component, or does not increase the first component, but reduce the second component. Degeneracy and Cap Decomposition do not increase the first component, but reduce the second component. The Variable Substitution rule reduces the first component: indeed, it replaces any variable x by a term not containing x, and also not containing any fresh variables. □

THEOREM 1. *Suppose* Γ *is a constraint set satisfying the Standard Protocol property. Then every* $\mathcal{I}_D$*-derivation from* Γ *is finite.*

PROOF. The well-founded measure $M(\Gamma)$ decreases at each inference. □

5.2 $\mathcal{I}_D$ is Sound and Complete

$\mathcal{I}_D$ is Sound: We only prove the soundness of the Homomorphic Deduction rule. The soundness of the other rules of $\mathcal{I}_D$ should be straightforward.

LEMMA 10. *Let Γ be a constraint set, and Γ' the constraint set derived from Γ by Homomorphic Deduction. If a substitution σ satisfies Γ', it also satisfies Γ.*

PROOF. We consider all the ground instances of the Homomorphic Deduction rule. We show that if there exists u such that
$u \in Cap(\{s_1, s_2, \ldots, s_n\}, \{p, \pi_1, \pi_2\})$, then
$e(u,t) \in Cap(\{e(s_1,t), e(s_2,t), \ldots, e(s_n,t)\}, SYM)$.
The proof is by induction on the structure of u. Base case: suppose u is s_i, for some $1 \leq i \leq n$, then the above statement is trivially true. In the general case, u can be made up from elements in $\{s_1, s_2, \ldots, s_n\}$ by using some caps in $\{p, \pi_1, \pi_2\}$, so we consider three cases for the inductive argument:

Case i): Suppose u is $p(u_1, u_2)$, where:
$e(u_1,t) \in Cap(\{e(s_1,t), e(s_2,t), \ldots, e(s_n,t)\}, SYM)$, and
$e(u_2,t) \in Cap(\{e(s_1,t), e(s_2,t), \ldots, e(s_n,t)\}, SYM)$.

We have to show that
$e(p(u_1,u_2),t) \in Cap(\{e(s_1,t), e(s_2,t), \ldots, e(s_n,t)\}, SYM)$.
Because $e(u_1,t)$ and $e(u_2,t)$ are both in the cap closure, we get that $p(e(u_1,t), e(u_2,t))$ is in the cap closure. Because of HE, $e(p(u_1,u_2),t)$ is also in the cap closure.

Case ii): Suppose u is $\pi_1(u_1)$, where $u_1 = p(w,w')$ for some ground terms w, w', and suppose
$e(u_1,t) \in Cap(\{e(s_1,t), e(s_2,t), \ldots, e(s_n,t)\}, SYM)$.
We want to show that:
$e(\pi_1(u_1),t) \in Cap(\{e(s_1,t), e(s_2,t), \ldots, e(s_n,t)\}, SYM)$.
In other words, we want to show that $e(w,t)$ is in the cap closure; this follows because $e(u_1,t) = e(p(w,w'),t)$ is in the cap closure, and because of HE.

Case iii): Suppose u is $\pi_2(u_1)$, where $u_1 = p(w,w')$ for some ground terms w, w', and suppose
$e(u_1,t) \in Cap(\{e(s_1,t), e(s_2,t), \ldots, e(s_n,t)\}, SYM)$. We conclude here exactly as in Case ii). □

PROPOSITION 1. *Let Γ_0 be a set of cap constraints satisfying the Standard Protocol property. Suppose $\Gamma_0, \Gamma_1, \ldots, \Gamma_n$ is an $\mathcal{I}_D$-derivation, and σ a substitution that satisfies Γ_n; then σ satisfies also Γ_0.*

PROOF. Suppose σ satisfies Γ_{i+1}; observe then that no inference rule, from step i to step $i+1$, adds any constraint that can be inconsistent with Γ_i; it follows that σ also satisfies Γ_i. □

$\mathcal{I}_D$ is Complete:.
We turn our attention now to showing that $\mathcal{I}_D$ is 'satisfiability-complete' for active deduction modulo HE; and for that, we need our theory to be convergent. So we extend HE by adding a single 'meta'-reduction rule, to get the following theory, that we denote as HE^+:

$\pi_1(p(x,y)) \rightarrow x$
$\pi_2(p(x,y)) \rightarrow y$
$d(e(x,y),y) \rightarrow x$
$e(p(x,y),z) \rightarrow p(e(x,z), e(y,z))$
$d(P_v(e(x_1,z), \cdots, e(x_n,z)), z) \rightarrow P_v(x_1, \cdots, x_n)$
where v is any legal bit string sequence of length n.

The following lemma is immediate:

LEMMA 11. *HE^+ is convergent, and is equivalent to HE.*

We observe next that the $\mathcal{E}_h$-Unifn algorithm of Section 4 is complete for solving cap constraints modulo HE^+ (or, equivalently: modulo HE):

LEMMA 12. *Let Γ be a set of cap constraints, s and t any two terms appearing in Γ, and let σ be a HE^+-normalized substitution. Then σ is a unifier of s and t modulo HE^+ if and only if σ is a unifier of s and t modulo $\mathcal{E}_h$.*

PROOF. HE^+ being convergent, we can solve $s = t$ modulo HE^+ by using narrowing. By Lemma 8, d, π_1, π_2 do not occur in s or t, so the narrowing process will only use the single rule of $\mathcal{E}_h$. □

In the sequel, we shall be needing the following notion of a *minimal solution* for a set of cap constraints Γ:

DEFINITION 8. *Given two solutions σ, τ for a set of cap constraints Γ, σ is said to be smaller than τ iff $|x\sigma| \leq |x\tau|$ for every variable x appearing in Γ.* (For any term t, its size is denoted $|t|$.) *A solution σ is* minimal *iff no solution is* strictly smaller *than σ.*

LEMMA 13. *Let $S \rhd_{(SYM,HE)} t$ be a cap constraint with the constructor property, that admits as minimal solution a ground substitution σ in HE^+-normal form. Then for any variable x, $x\sigma$ cannot contain the symbols d, π_1, π_2.*

PROOF. By assumption, there is a term $u \in Cap(S, SYM)$ such that $u\sigma \leftrightarrow_{HE^+} t\sigma$. Since HE^+ is convergent, there exists then a ground term w, and a HE^+-rewrite proof: $u\sigma \longrightarrow^*_{HE^+} w \longleftarrow^*_{HE^+} t\sigma$. Now, by the constructor property, the only function symbols that can appear in t, and in the terms of S, are 'p' and 'e'. Suppose some of these terms contained a variable x such that $x\sigma$ contains one of the symbols d, π_1, π_2. Since σ is assumed to be in HE^+-normal form, it follows that none of the innermost subterms of $u\sigma$ or $t\sigma$ with d, π_1, or π_2 as top symbols can be reducible. So, such terms will remain as they are all along the rewrite proof; they can then be replaced by arbitrarily chosen constants, and we will get a solution for the constraint that is strictly smaller than σ – contradicting the minimality assumption on σ (cf. Definition 8).

A similar reasoning works also when the rewrite proof between $u\sigma$ and $t\sigma$ is trivial, i.e., $u\sigma$ and $t\sigma$ are identical: suppose, for instance, that a term with root symbol in $\{d, \pi_1, \pi_2\}$ appears at certain positions in the term $u\sigma$; then all these positions will have to be in the substitution σ; one can then construct (as above) a σ' strictly smaller than σ, such that $u\sigma'$ and $t\sigma'$ are identical. □

THEOREM 2. *Let Γ be a satisfiable set of cap constraints over HE, with the Standard Protocol Property. Then there is an $\mathcal{I}_D$-derivation $\Gamma_0, \Gamma_1, \ldots, \Gamma_n$, such that: (i) Γ_0 is Γ, (ii) each Γ_i is satisfiable, and (iii) Γ_n is empty.*

PROOF. We actually show the following: Let Γ be any satisfiable, non-empty set of cap constraints over HE (with the Standard Protocol Property), and σ any given minimal ground solution for Γ; then there is an $\mathcal{I}_D$-inference $\Gamma \Rightarrow_{\mathcal{I}_D} \Gamma'$ such that σ induces a solution for the set of cap constraints Γ'. For this, we may assume that σ is a HE^+-*normalized* substitution.

Case 1: Consider first the case, where for *some* constraint $S \rhd_{(SYM,HE)} u$ in Γ, the set S contains a term with 'p' on top. In this case, the Projection inference of $\mathcal{I}_D$ can be applied; it is easy then to derive from σ a solution to the inferred cap constraint set Γ', which proves our claim.

So we may assume henceforth, that for *every* cap constraint $S \rhd_{(SYM,HE)} u$ in Γ, each term in S is *either* a pure variable or a constant, *or* has 'e' as its top symbol. (Note: by Lemma 8, the symbols d, π_1, π_2 cannot appear in Γ.)

Case 2: Suppose then that for *some* cap constraint $S \rhd u$ in Γ, the set S contains *only terms with* 'e' on top, and u is neither a variable nor a constant. We have two subcases here:

- case 2.1: Either the top symbol of the RHS term u of the constraint is 'p': in which case, a Cap Decomposition inference is applicable to Γ;
- case 2.2: Or else the top symbol of the RHS term u is 'e': in this subcase, (depending on the symbol set SYM) either Homomorphic Deduction or Decryption can be applied Γ.

In both subcases, one can derive from σ a solution for the inferred system Γ'.

Case 3: So, we assume henceforth, that in *every* cap constraint $S \rhd_{(SYM,HE)} u$ in Γ, pure variables or constants have to appear as members of S (the RHS u can be a variable, or a constant, or a term with 'p' or 'e' on top).

Now, suppose Γ contains a constraint $S \rhd_{(SYM,HE)} u$ where the term u has 'p' or 'e' as top symbol; in such a case, one of the Degeneracy, Cap Decomposition, Decryption or Homomorphic Deduction inference rules can be applied to prove our claim. So we may assume that *the RHS of* all *constraints in Γ are either pure variables or constants.* We have then two subcases to consider:

- Case 3.1: Suppose the RHS of *all* constraints in Γ are constants. In this case, the terms to the left of *any* constraint in Γ must all be ground terms, because of the Variable Introduction property. In this case, the Degeneracy rule applies and proves our claim.

- Case 3.2: So remains finally to consider the case where for some cap constraint in Γ, the RHS term is a pure variable.

Let $S_1 \rhd_{(SYM,HE)} z_1, \cdots, S_r \rhd_{(SYM,HE)} z_r$ be *all the* cap constraints in Γ, whose RHS are pure variables. For any $j, 1 \leq j \leq r$, let the *e-height* of z_j (wrt the given solution σ) be defined as the number of times the symbol 'e' appears in the ground term $z_j\sigma$; we then choose the first index $p \in \{1, \ldots, r\}$ for which the e-height of z_p is *maximal*. It follows that z_p cannot appear at a *non-root position* in any of the terms in any of the sets $S_j, 1 \leq j \leq r$: indeed, due to our previous reductions, any term in an S_j that contains z_p at a non-root position would have 'e' as top symbol, and the corresponding RHS variable z_j would be of a strictly bigger e-height. In other words, if z_p appears in any of the S_j, then it is a member by itself of the set S_j.

Now, by assumption the substitution σ solves each constraint in Γ, and in particular $S_p \rhd_{(SYM,HE)} z_p$, with p as chosen above; by Definition 4, this implies that $z_p\sigma$ is equal, modulo HE, to some term $t \in Cap(\{t_1\sigma, \ldots, t_m\sigma\}, SYM)$, where the $t_1, \ldots, t_m$ are some non-variable terms (not containing z_p) over the LHS of the constraints in Γ; and this term t must also be in $Cap(S\sigma, SYM)$ for every constraint in Γ of the form $S \rhd_{(SYM,HE)} z_p$, i.e., whose RHS is z_p.

We then define a substitution $\theta : z_p \mapsto P_v(t_1, \cdots, t_m)$, where v is some legal bit string sequence of length m, and apply the Variable Substitution inference to Γ, wrt θ. The system thus inferred $\Gamma' = \Gamma\theta$ has a smaller number of variables: $Vars(\Gamma') = Vars(\Gamma) \setminus \{z_p\}$. From the given minimal ground solution σ for Γ, we then derive a ground substitution σ' on the set of variables of Γ', by setting $y\sigma' = y\sigma$, for every variable y of Γ'.

The claim is that σ' extends naturally to a solution for the constraint system Γ'. To prove this, note first that the RHS of any constraint in Γ' is either a variable $\neq z_p$ of Γ, or is the term $P_v(t_1, \cdots, t_m)$. It is therefore obvious that σ' solves any constraint in Γ' whose RHS is a variable; so, we only need to consider a constraint in Γ' of the form $S' \rhd_{(SYM,HE)} P_v(t_1, \cdots, t_m)$, with $S' = S\theta$, obtained by applying θ to a constraint in Γ of the form $S \rhd_{(SYM,HE)} z_p$.

To prove our current claim, we may suppose $z_p \notin S$, or $P_v(t_1, \cdots, t_m) \notin S'$. Therefore $S\sigma = S'\sigma'$, and our above term $t \in Cap(\{t_1\sigma, \ldots, t_m\sigma\}, SYM)$ is also in the cap closure $Cap(S'\sigma', SYM)$. Now, the symbols p, π_1, π_2 are in SYM in all cases; on the other hand, by Lemma 13, they do *not* appear in t; these facts put together allow us to conclude that σ' does induce a solution also for the cap constraint $S' \rhd_{(SYM,HE)} P_v(t_1, \cdots, t_m)$ of the inferred system Γ'. □

5.3 Illustrative Example - Contd.

In Section 3, we discussed the "NEEDHAM-SCHROEDER SYMMETRIC KEY PROTOCOL" and the attack on it based on homomorphism; we also saw how we generated the following cap constraints.

i. $\{A, B, N_a\} \rhd_{(\{p,\pi_1,\pi_2,e,d\},HE)} p(p(A, B), N_a)$
ii. $\{A, B, N_a, e(p(p(p(N_a, B), K_{ab}), e(p(K_{ab}, A), K_{bs})), K_{as})\}$
$\rhd_{(\{p,\pi_1,\pi_2,e,d\},HE)} p(e(p(x, B), K_{as}), x)$

To solve (i), we use Cap Decomposition twice and get the following constraints; each of them can be solved using Degeneracy.

i-1. $\{A, B, N_a\} \triangleright_{(\{p,\pi_1,\pi_2,e,d\},HE)} A$
i-2. $\{A, B, N_a\} \triangleright_{(\{p,\pi_1,\pi_2,e,d\},HE)} B$
i-3. $\{A, B, N_a\} \triangleright_{(\{p,\pi_1,\pi_2,e,d\},HE)} N_a$

To solve (ii), first we use Cap Decomposition:
Deduced constraint ii-1.
$\{A, B, N_a, e(p(p(p(N_a, B), K_{ab}), e(p(K_{ab}, A), K_{bs})), K_{as})\}$
$\triangleright_{(\{p,\pi_1,\pi_2,e,d\},HE)} x$
Deduced constraint ii-2.
$\{A, B, N_a, e(p(p(p(N_a, B), K_{ab}), e(p(K_{ab}, A), K_{bs})), K_{as})\}$
$\triangleright_{(\{p,\pi_1,\pi_2,e,d\},HE)} e(p(x, B), K_{as})$

Now, ii-1 can be solved by using Degeneracy: set x to N_a. And ii-2 can be solved by using Homomorphic Deduction, Projection and Degeneracy:

(Homomorphic Deduction)
$\{p(p(p(N_a, B), K_{ab}), e(p(K_{ab}, A), K_{bs}))\}$
$\triangleright_{(\{p,\pi_1,\pi_2\},HE)} p(x, B)$
(Projection)
$\{p(p(N_a, B), K_{ab}), e(p(K_{ab}, A), K_{bs})\}$
$\triangleright_{(\{p,\pi_1,\pi_2\},HE)} p(x, B)$
$\{p(N_a, B), K_{ab}), e(p(K_{ab}, A, K_{bs})\}$
$\triangleright_{(\{p,\pi_1,\pi_2\},HE)} p(x, B)$
(Degeneracy) $\emptyset$

6. CONCLUSION

There are two basic reasons why the approach presented in this paper works for active deduction modulo Homomorphic Encryption. One is that normalized narrowing wrt the system HE^+ (equivalent to HE) terminates; and the other is that such a narrowing allows us to 'reduce' deduction modulo HE^+ to deduction modulo the subtheory $\mathcal{E}_h$, for which unification is essentially syntactic. It would be of interest to try to generalize our approach to other algebraic intruder theories for which similar reductions are possible, and to cases where the encryption schemes are assumed to satisfy certain group homomorphism properties. We also would like to mention a couple of other points:

i) The form of the rules of our rewrite system HE might lead to conclude that our approach would work only for symmetric encryption schemes; but it is not hard to adapt the system HE and the approach to handle asymmetric keys.

ii) The approach presented is appropriate for encryption based on ECB (Electronic Code Book) block chaining. A block chaining technique less vulnerable than ECB is CBC (Cipher Block Chaining); cf. e.g., [16]. Some works ([10]) have considered a version of homomorphism theory that is incomplete for such an encryption, i.e., might miss some attacks. The following convergent AC-rewrite system R_1 (where $+$ = XOR is AC) models such an encryption:

$x + 0 \to x$, $\quad x + x \to 0$
$p_1(cons(x, y)) \to x$, $\quad p_2(cons(x, y)) \to y$
$dec(enc(x, y), y) \to x$
$cbc(cons(x, y), z, w) \to$
$\quad cons(\, enc(z + x, w), cbc(y, enc(z + x, w), w)\,)$
$cbc(nil, z, k) \to nil$

Here $cbc(ls, vs, k)$ stands for the encryption, with k as key, of the list ls of message blocks, with vs as padding vector; while $enc(m, k)$ (resp. $dec(m, k)$) stands for message block m encrypted (resp. decrypted) with key k. Passive deduction modulo R_1 can be shown to be decidable, by extending the results of [1] to AC-rewriting. Refining the approach of our current paper, into one that would be complete for active deduction modulo R_1, is part of ongoing work.

In this paper we have given an algorithm for a homomorphic operator over a free theory, which models the ECB encryption algorithm. This is just a first step in modeling encryption algorithms. We hope to extend our result to homomorphic operators over more expressive theories, and more useful for cryptographic protocol analysis.

7. REFERENCES

[1] S. Anantharaman, P. Narendran, M. Rusinowitch. "Intruders with Caps". In *Proc. of Int. Conf. RTA'07*, LNCS 4533, pp. 20–35, Springer-Verlag, June 2007.

[2] S. Anantharaman, H. Lin, C. Lynch, P. Narendran, M. Rusinowitch. "Unification modulo Homomorphic Encryption". In *Proc. of Int. Conf. FROCOS 2009*, TRENTO-Italy, LNAI 5749, pp. 100–116, Springer-Verlag, September 2009.

[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, L. Viganò. "An On-The-Fly Model-Checker for Security Protocol Analysis". In *Proc. of ESORICS'03*, LNCS 2808, pp. 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*, pp. 16–25, 2005.

[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 *Proc. of the Logic In Computer Science Conference, LICS'03*, pp. 261–270, 2003.

[8] H. Comon-Lundh, R. Treinen. "Easy Intruder Deductions." Verification: Theory and Practice In *LNCS* 2772, pp. 225–242, Springer-Verlag, February 2004.

[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] V. Cortier, M. Rusinowitch, E. Zalinescu. "A resolution strategy for verifying cryptographic protocols with CBC encryption and blind signatures". In *Proc. of the 7th ACM SIGPLAN Symposium PPDP 2005*, pp. 12–22.

[11] S. Delaune, F. Jacquemard. "A decision procedure for the verification of security protocols with explicit

destructors". In *Proc. of the 11th ACM Conference on Computer and Communications Security (CCS'04)*, pp. 278–287, Washington, D.C., USA, October 2004. ACM Press.

[12] S. Delaune, P. Lafourcade, D. Lugiez, R. Treinen, "Symbolic protocol analysis for monoidal equational theories." In *Information and Computation* 206(2-4), pp. 312-351, 2008.

[13] S. Escobar, C. Meadows, J. Meseguer. "A Rewriting-Based Inference System for the NRL Protocol Analyzer and its Meta-Logical Properties." In *Theoretical Computer Science, Vol. 367(1-2)*, pp. 162–202, 2006.

[14] F. J.-T. Fabrega, J. C. Herzog, J. D. Guttman. "Strand Spaces: Why is a Security Protocol Correct?" In *Proc. of IEEE Symposium on Security and Privacy*, May 1998.

[15] J.M. Hullot. "Canonical Forms and Unification". In *Proc. of the Int. Conf. on Automated Deduction CADE-5*, LNCS 87, pp. 318–334, Springer-Verlag, 1980.

[16] S. Kremer, M. D. Ryan. "Analysing the vulnerability of protocols to produce known-pair and chosen-text attacks". In *Proc. of the 2nd Int. Workshop on Security Issues in Coordination Models, Languages, and Systems (SecCo 2004)*, ENTCS, Vol. 128, Issue 5, pp. 87–104, 2004.

[17] C. Meadows, 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.

[18] J. Millen, V. Shmatikov. "Constraint Solving for Bounded-Process Cryptographic Protocol Analysis" In *Proc. of the 8th ACM Conference on Computer and Communications Security* pp. 166–175, 2001.

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

[20] C. Weidenbach. "Towards an automatic analysis of security protocols". In *Proc. of the 16th Int. Conf. on Automated Deduction, CADE-16*, LNAI 1632, pp. 378–382, Springer-Verlag, 1999.