Unification modulo Homomorphic Encryption

S. Anantharaman[1], H. Lin, C. Lynch,
P. Narendran, M. Rusinowitch

[1] Laboratoire d'Informatique Fondamentale d'Orléans (LIFO)
Université d'Orléans (France)

The Outline

1. Motivation: Analyzing Crypto-Protocols
2. Spec HE for Homomorphic Encryption
3. Unification modulo HE
   - The Method
   - Examples
4. Conclusive Remarks

Motivation

A method for analyzing Crypto-Protocols:

R = convergent TRS modeling 'Intruder' abilities
$\mathcal{C}$ = set of Horn clauses (constraints) modeling protocol steps
G = Intruder's initial knowledge

Motivation

A method for analyzing Crypto-Protocols:

R = convergent TRS modeling 'Intruder' abilities
$\mathcal{C}$ = set of Horn clauses (constraints) modeling protocol steps
G = Intruder's initial knowledge

Cap-closure(G) = Evolution of Intruder's knowledge:
Computed using R-Narrowing and (usual or R-) Unification, with, or without, active interaction with protocol steps

Motivation

A method for analyzing Crypto-Protocols:

R = convergent TRS modeling 'Intruder' abilities
$\mathcal{C}$ = set of Horn clauses (constraints) modeling protocol steps
G = Intruder's initial knowledge

Cap-closure(G) = Evolution of Intruder's knowledge:
Computed using R-Narrowing and (usual or R-) Unification,
with, or without, active interaction with protocol steps

Secrecy Attack:
A certain ground term m – intended secret for Intruder –
is in the Cap-closure

Motivation

A method for analyzing Crypto-Protocols:

R = convergent TRS modeling 'Intruder' abilities
$\mathcal{C}$ = set of Horn clauses (constraints) modeling protocol steps
G = Intruder's initial knowledge

Cap-closure(G) = Evolution of Intruder's knowledge:
Computed using R-Narrowing and (usual or R-) Unification, with, or without, active interaction with protocol steps

Secrecy Attack:
A certain ground term m – intended secret for Intruder – is in the Cap-closure

Authentication Attack:
A certain "frame of Cap-constraints" is satisfiable

Motivation - contd.

Each attack formulable as a Deduction Problem: Solving a set of Cap-constraints in terms of R-Unification and/or R-narrowing

Deduction Active, or Passive:
Intruder participates, or not, in the Protocol

Motivation - contd.

Each attack formulable as a Deduction Problem: Solving a set of Cap-constraints in terms of R-Unification and/or R-narrowing

Deduction Active, or Passive:

Intruder participates, or not, in the Protocol

Decidability of R-Unification: Necessary for Active Deduction

Motivation - contd.

Each attack formulable as a Deduction Problem: Solving a set of Cap-constraints in terms of R-Unification and/or R-narrowing

Deduction Active, or Passive:
Intruder participates, or not, in the Protocol

Decidability of R-Unification: Necessary for Active Deduction

Deduction problems studied by many:
Baudet, Abadi-Cortier, Comon-Treinen, Comon-Shmatikov, Rusinowitch-Turuani, Rusinowitch-Chevalier, ...

Motivation - contd.

Each attack formulable as a Deduction Problem: Solving a set of Cap-constraints in terms of R-Unification and/or R-narrowing

Deduction Active, or Passive:
Intruder participates, or not, in the Protocol

Decidability of R-Unification: Necessary for Active Deduction

Deduction problems studied by many:
Baudet, Abadi-Cortier, Comon-Treinen, Comon-Shmatikov, Rusinowitch-Turuani, Rusinowitch-Chevalier, ...

Algos obtained, mostly when:
- R is pure: RHS of each rule in R is a variable
- R is dwindling (subterm property):
  each RHS is subterm of corresponding LHS

Motivation - contd.

S.A, Narendran, Rusinowitch (RTA'07): A complete Algoritm for Passive Deduction, if *R* is dwindling or "Delta-Strong"

Motivation - contd.

S.A, Narendran, Rusinowitch (RTA'07): A complete Algoritm for Passive Deduction, if R is dwindling or "Delta-Strong"

Algorithm does not resort to R-unification; holds for a non-dwindling convergent TRS HE specifying Homomorphic Encryption

Motivation - contd.

S.A, Narendran, Rusinowitch (RTA'07): A complete Algoritm for Passive Deduction, if R is dwindling or "Delta-Strong"

Algorithm does not resort to R-unification; holds for a non-dwindling convergent TRS HE specifying Homomorphic Encryption

We are thus led to:

Question: Is Unification modulo HE decidable?

Spec HE for Homomorphic Encryption

The following rewrite system HE is convergent:

$$
\begin{aligned}
p_1(x.y) &\rightarrow x \\
p_2(x.y) &\rightarrow y \\
enc(dec(x, y), y) &\rightarrow x \\
dec(enc(x, y), y) &\rightarrow x \\
enc(x.y, z) &\rightarrow enc(x, z).enc(y, z) \\
dec(x.y, z) &\rightarrow dec(x, z).dec(y, z)
\end{aligned}
$$

Spec HE for Homomorphic Encryption

The following rewrite system HE is convergent:

$$\begin{array}{rcl} p_1(x.y) & \rightarrow & x \\ p_2(x.y) & \rightarrow & y \\ enc(dec(x,y),y) & \rightarrow & x \\ dec(enc(x,y),y) & \rightarrow & x \\ enc(x.y,z) & \rightarrow & enc(x,z).enc(y,z) \\ dec(x.y,z) & \rightarrow & dec(x,z).dec(y,z) \end{array}$$

HE *models Homomorphic Encryption in the following case*:

'$\cdot$' stands for 'pairing' message-blocks; $enc(x,y)$ is message x encrypted with key y, $dec(x,y)$ is message x decrypted with key y.

enc (resp. *dec*) stands for ciphering (resp. deciphering) 'fixed-size' message-blocks (ECB = Electronic Code Book)

Comments on HE

For any key y defines a homomorphism on terms, wrt pairing '$\cdot$':
$h_y(x) = enc(x, y)$, which admits as inverse $\overline{h}_y(x) = dec(x, y)$

Remark 1. HE-Unifn does **not** reduce to Unifn modulo 1-sided distributivity (of Tiden-Arnborg), due to existence of inverse homomorphisms.

Comments on HE

For any key y defines a homomorphism on terms, wrt pairing '$\cdot$':
$h_y(x) = enc(x, y)$, which admits as inverse $\overline{h}_y(x) = dec(x, y)$

Remark 1. HE-Unifn does **not** reduce to Unifn modulo 1-sided distributivity (of Tiden-Arnborg), due to existence of inverse homomorphisms.

Remark 2. The reasonings we develop hold also for some other specs for homomorphic encryption, not using an explicit decryption function (e.g., decrypt = encrypt with inverse key. cf. Concluding Section).

In particular, they are easily adapted to the case of asymmetric keys.

Unification modulo HE

We assume: Pur HE-Unifn problems $\mathcal{P}$ are in *standard form*, with the following type of equations (where X, Y, Z, T are variables, a is any ground constant):

- Pairings: $Z = X.Y$
- Equations of *enc*-type: $Z = enc(X, T) = h_T(X)$
- Equalities: $Z = T$ or $Z = a$

Unification modulo HE

We assume: Pur HE-Unifn problems $\mathcal{P}$ are in *standard form*, with the following type of equations (where X, Y, Z, T are variables, a is any ground constant):

- Pairings: $Z = X.Y$
- Equations of *enc*-type: $Z = enc(X, T) = h_T(X)$
- Equalities: $Z = T$ or $Z = a$

$\mathcal{X}_\mathcal{P}$ = set of all variables of $\mathcal{P}$
$\mathcal{K}_\mathcal{P}$ = set of all key variables (and key constants) of $\mathcal{P}$
$\mathcal{H} = \mathcal{H}_\mathcal{P}$ = set of all homomorphisms (and their inverses) defined by the key variables/constants of $\mathcal{P}$.

Unification modulo HE - contd.

Dependency graph $G = G_{\mathcal{P}}$ for $\mathcal{P}$:
Nodes = the variables (or constants) of $\mathcal{P}$

Unification modulo HE - contd.

Dependency graph $G = G_{\mathcal{P}}$ for $\mathcal{P}$:
Nodes = the variables (or constants) of $\mathcal{P}$

From node Z to node X on G, there is an oriented arc on G iff:
- $\mathcal{P}$ has an equation $Z = h_Y(X)$ (resp. $X = h_Y(Z)$), for some Y:
  then label the arc with h_Y (resp. $\overline{h}_Y$)
- $\mathcal{P}$ has an equation $Z = X.V$ (resp. $Z = V.X$):
  then label the arc with p_1 (resp. with p_2).

Unification modulo HE - contd.

Dependency graph $G = G_{\mathcal{P}}$ for $\mathcal{P}$:
Nodes = the variables (or constants) of $\mathcal{P}$

From node Z to node X on G, there is an oriented arc on G iff:
- $\mathcal{P}$ has an equation $Z = h_Y(X)$ (resp. $X = h_Y(Z)$), for some Y:
  then label the arc with h_Y (resp. $\overline{h}_Y$)
- $\mathcal{P}$ has an equation $Z = X.V$ (resp. $Z = V.X$):
  then label the arc with p_1 (resp. with p_2).

No 'equality arc' on the graph $G_{\mathcal{P}}$.

Unification modulo HE - contd.

Dependency graph $G = G_{\mathcal{P}}$ for $\mathcal{P}$:
Nodes = the variables (or constants) of $\mathcal{P}$

From node Z to node X on G, there is an oriented arc on G iff:
- $\mathcal{P}$ has an equation $Z = h_Y(X)$ (resp. $X = h_Y(Z)$), for some Y:
  then label the arc with h_Y (resp. $\overline{h}_Y$)
- $\mathcal{P}$ has an equation $Z = X.V$ (resp. $Z = V.X$):
  then label the arc with p_1 (resp. with p_2).

No 'equality arc' on the graph $G_{\mathcal{P}}$.

Semantics: If G contains an edge $Z \to^h X$, with $h \in \mathcal{H}$,
then Z evaluable by applying homomorphism h to the evaluation of X.

Unification modulo HE: the Method

Let **Pair**, **Eq**, **Enc**, denote respectively:

the set of pairings, equalities, and *enc*-equations of $\mathcal{P}$.

Observation 1: If **Enc** $= \emptyset$ then $\mathcal{P}$ is easily solved.
So always assume the presence of *enc*-equations.

Unification modulo HE: the Method

Let **Pair**, **Eq**, **Enc**, denote respectively:

the set of pairings, equalities, and *enc*-equations of $\mathcal{P}$.

Observation 1: If **Enc** $= \emptyset$ then $\mathcal{P}$ is easily solved.
So always assume the presence of *enc*-equations.

• *Suppose the following two properties* - denoted (#) - *hold*:

- G is irredundant:

  V, W distinct nodes on $G \;\Rightarrow\; V \neq_{\mathbf{Eq}} W$

- Every node Z on G is '**non-critical**':

  If there is an outgoing h- or $\overline{h}$- arc from Z on G,
  then there's NO outgoing p_1- or p_2- arc from Z.

Unification modulo HE: the Method

Let **Pair**, **Eq**, **Enc**, denote respectively:
the set of pairings, equalities, and *enc*-equations of $\mathcal{P}$.

Observation 1: If $\textbf{Enc} = \emptyset$ then $\mathcal{P}$ is easily solved.
So always assume the presence of *enc*-equations.

- *Suppose the following two properties* - denoted (#) - *hold*:

- G is irredundant:
  V, W distinct nodes on $G \;\Rightarrow\; V \neq_{\textbf{Eq}} W$

- Every node Z on G is '**non-critical**':
  If there is an outgoing h- or $\overline{h}$- arc from Z on G,
  then there's NO outgoing p_1- or p_2- arc from Z.

- And suppose also, we know then how to solve the subproblem $\mathcal{P}'$ of $\mathcal{P}$ formed of its *enc*-equations.

Then we can solve $\mathcal{P}$ by combining solution for $\mathcal{P}'$
with (solution for) the pairings and equalities of $\mathcal{P}$.

Unification modulo HE: the Method

Method for solving any HE-Unifn problem $\mathcal{P}$ in standard form:

Transform $\mathcal{P}$ into an equivalent problem $\mathcal{P}_1$ such that the depenedency graph of $\mathcal{P}_1$ has the above properties.

Unification modulo HE: the Method

Method for solving any HE-Unifn problem $\mathcal{P}$ in standard form:

> Transform $\mathcal{P}$ into an equivalent problem $\mathcal{P}_1$ such that the depenedency graph of $\mathcal{P}_1$ has the above properties.

Example 1: Consider the problem $\mathcal{P}$:

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

Its graph is irredundant, but node Y is 'crtitical' (= not non-critical).
We transform $\mathcal{P}$, by reasoning as follows:

Unification modulo HE: the Method

Method for solving any HE-Unifn problem $\mathcal{P}$ in standard form:

Transform $\mathcal{P}$ into an equivalent problem $\mathcal{P}_1$ such that the depenedency graph of $\mathcal{P}_1$ has the above properties.

Example 1: Consider the problem $\mathcal{P}$:

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

Its graph is irredundant, but node Y is 'crtitical' (= not non-critical).
We transform $\mathcal{P}$, by reasoning as follows:

Since Y is a pair, Z must also be a pair

So we *split* Z as $Z = Z_1.Z_2$, introducing fresh vars Z_1, Z_2.

Unification modulo HE: the Method

Method for solving any HE-Unifn problem $\mathcal{P}$ in standard form:

Transform $\mathcal{P}$ into an equivalent problem $\mathcal{P}_1$ such that the depenedency graph of $\mathcal{P}_1$ has the above properties.

Example 1: Consider the problem $\mathcal{P}$:

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

Its graph is irredundant, but node Y is 'crtitical' (= not non-critical).
We transform $\mathcal{P}$, by reasoning as follows:

Since Y is a pair, Z must also be a pair

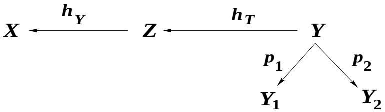
So we *split* Z as $Z = Z_1.Z_2$, introducing fresh vars Z_1, Z_2.

We also need to split X subsequently, for the same reason.

The problem and its graph evolve, as follows
(where, for readability the $\overline{h}$-arcs are not put in):

HE-Unification: the Method (contd.)

Initial graph:



HE-Unification: the Method (contd.)

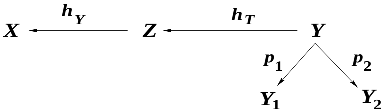
Initial graph:

$$\begin{tikzcd}
X & & Z \arrow[ll, "h_Y"'] & & & Y \arrow[lll, "h_T"'] \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\
& & & & Y_1 & & Y_2
\end{tikzcd}$$

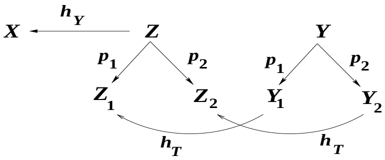
Split Z:

$$\begin{tikzcd}
X & & Z \arrow[ll, "h_Y"'] \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & & & Y \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\
& Z_1 & & Z_2 & Y_1 \arrow[lll, bend left, "h_T"] & & Y_2 \arrow[lll, bend left, "h_T"]
\end{tikzcd}$$

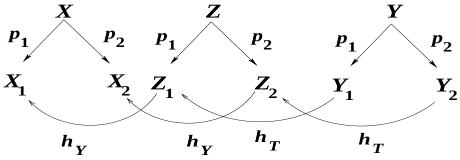
Initial graph:



Split Z:



Split Z and X:



HE-Unification: the Method (contd.)

No node on the final graph is critical, so we stop.
It is the graph of the following problem $\mathcal{P}_1$:

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

HE-Unification: the Method (contd.)

No node on the final graph is critical, so we stop.
It is the graph of the following problem $\mathcal{P}_1$:

$$Z_1 = enc(X_1, Y), \quad Z_2 = enc(X_2, Y),$$
$$Y_1 = enc(Z_1, T), \quad Y_2 = enc(Z_2, T),$$
$$Y = Y_1.Y_2, \; Z = Z_1.Z_2, \; X = X_1.X_2.$$

We solve – without difficulty in this case – the subproblem $\mathcal{P}_1'$ of $\mathcal{P}_1$ formed of its four *enc*-equations.

We do it in a “lazy” style: No variable is instantiated unless necessary.

HE-Unification: the Method (contd.)

No node on the final graph is critical, so we stop.
It is the graph of the following problem $\mathcal{P}_1$:

$$Z_1 = enc(X_1, Y), \quad Z_2 = enc(X_2, Y),$$
$$Y_1 = enc(Z_1, T), \quad Y_2 = enc(Z_2, T),$$
$$Y = Y_1.Y_2, \ \ Z = Z_1.Z_2, \ \ X = X_1.X_2.$$

We solve – without difficulty in this case – the subproblem $\mathcal{P}_1'$ of $\mathcal{P}_1$ formed of its four *enc*-equations.

We do it in a “lazy” style: No variable is instantiated unless necessary.

We thus solve for $\mathcal{P}_1'$ here, as:

$$Z_1 = h_Y(X_1), \ Z_2 = h_Y(X_2), \ Y_1 = h_T(Z_1), \ Y_2 = h_T(Z_2)$$

leaving Y, T, X_1, X_2 uninstantiated.

HE-Unification: the Method (contd.)

No node on the final graph is critical, so we stop.
It is the graph of the following problem $\mathcal{P}_1$:

$$Z_1 = enc(X_1, Y), \quad Z_2 = enc(X_2, Y),$$
$$Y_1 = enc(Z_1, T), \quad Y_2 = enc(Z_2, T),$$
$$Y = Y_1.Y_2, \quad Z = Z_1.Z_2, \quad X = X_1.X_2.$$

We solve – without difficulty in this case – the subproblem $\mathcal{P}'_1$ of $\mathcal{P}_1$ formed of its four *enc*-equations.

We do it in a “lazy” style: No variable is instantiated unless necessary.

We thus solve for $\mathcal{P}'_1$ here, as:

$$Z_1 = h_Y(X_1), \; Z_2 = h_Y(X_2), \; Y_1 = h_T(Z_1), \; Y_2 = h_T(Z_2)$$

leaving Y, T, X_1, X_2 uninstantiated.

We end up by solving for Y, Z and X,
by combining this solution for $\mathcal{P}'_1$ with the pairings of $\mathcal{P}_1$

HE-Unification: the Method (contd.)

Thus, our method for solving an HE-Unifn problem $\mathcal{P}$ consists in three essential steps:

Step 1. Transform $\mathcal{P}$ into an equivalent $\mathcal{P}_1$ such that the graph $G_{\mathcal{P}_1}$ satisfies the two properties (#)

Step 2. Solve the subproblem $\mathcal{P}_1'$ formed of the *enc*-equations of $\mathcal{P}_1$.

Step 3. Solve $\mathcal{P}_1$: combine solution for $\mathcal{P}_1'$ with pairings and equalities of $\mathcal{P}_1$.

HE-Unification: the Method (contd.)

Thus, our method for solving an HE-Unifn problem $\mathcal{P}$ consists in three essential steps:

Step 1. Transform $\mathcal{P}$ into an equivalent $\mathcal{P}_1$ such that the graph $G_{\mathcal{P}_1}$ satisfies the two properties (#)

Step 2. Solve the subproblem $\mathcal{P}_1'$ formed of the *enc*-equations of $\mathcal{P}_1$.

Step 3. Solve $\mathcal{P}_1$: combine solution for $\mathcal{P}_1'$ with pairings and equalities of $\mathcal{P}_1$.

Definition:
(i) A problem $\mathcal{P}$ is *admissible* iff $G_{\mathcal{P}}$ satisfies properties (#)
(ii) $\mathcal{P}$ is *simple* iff $\mathcal{P}$ is admissible and has no 'pairings'
(iii) *Kernel* of an admissible problem $\mathcal{P}$
= the simple subproblem of $\mathcal{P}$ formed of its *enc*-equations.

The Method - Step 1

Guiding principles for transforming a problem $\mathcal{P}$ in standard from, into equivalent an admissible problem:

- Perfect Encryption:
  $(Z = enc(X, Y) \in \mathcal{P} \ \wedge \ Z = enc(X, Y') \in \mathcal{P}) \ \Rightarrow \ (Y = Y' \in \mathcal{P})$
  $(Z = enc(X, Y) \in \mathcal{P} \ \wedge \ Z = enc(X', Y) \in \mathcal{P}) \ \Rightarrow \ (X = X' \in \mathcal{P})$
- Pairing is free in HE:
  $(Z = X.Y \in \mathcal{P} \ \wedge \ Z = X'.Y' \in \mathcal{P}) \ \Rightarrow \ (X = X' \in \mathcal{P} \ \wedge \ Y = Y' \in \mathcal{P})$
- Split on Pairs:
  If $Z = enc(X, Y) \in \mathcal{P}$ and either Z or X splits as a pair, then the other must split too (intoduce fresh vars if necessary
- Keep the graph G of $\mathcal{P}$ irredundant:
  Distinct nodes Z', Z'' on G must not be equal modulo $=_{\mathbf{Eq}}$

The Method - Step 1

Guiding principles for transforming a problem $\mathcal{P}$ in standard from, into equivalent an admissible problem:

- Perfect Encryption:

  $(Z = enc(X, Y) \in \mathcal{P} \ \wedge \ Z = enc(X, Y') \in \mathcal{P}) \ \Rightarrow \ (Y = Y' \in \mathcal{P})$
  $(Z = enc(X, Y) \in \mathcal{P} \ \wedge \ Z = enc(X', Y) \in \mathcal{P}) \ \Rightarrow \ (X = X' \in \mathcal{P})$

- Pairing is free in HE:

  $(Z = X.Y \in \mathcal{P} \ \wedge \ Z = X'.Y' \in \mathcal{P}) \ \Rightarrow \ (X = X' \in \mathcal{P} \ \wedge \ Y = Y' \in \mathcal{P})$

- Split on Pairs:

  If $Z = enc(X, Y) \in \mathcal{P}$ and either Z or X splits as a pair, then the other must split too (intoduce fresh vars if necessary

- Keep the graph G of $\mathcal{P}$ irredundant:

  Distinct nodes Z', Z'' on G must not be equal modulo $=_{\mathbf{Eq}}$

These Principles are
- sound from the viewpoint of unification
- also meaningful cryptographically

Inference System for Step 1

The main Inference rules (the *Trimming* rules), on how **Eq**, **Pair**, **Enc** evolve under transformation, are based on these principles

Also an 'Occur-Check' inference: leads to Failure for easy cases of unsolvability; such as when $Z = enc(X, T) \in \mathcal{P}$ and $Z = X.Y \in \mathcal{P}$

Couple of other Failure rules:
- cases of clash between ground constants and/or pairings in $\mathcal{P}$

Inference System for Step 1

The main Inference rules (the *Trimming* rules), on how **Eq**, **Pair**, **Enc** evolve under transformation, are based on these principles

Also an 'Occur-Check' inference: leads to Failure for easy cases of unsolvability; such as when $Z = enc(X, T) \in \mathcal{P}$ and $Z = X.Y \in \mathcal{P}$

Couple of other Failure rules:
- cases of clash between ground constants and/or pairings in $\mathcal{P}$

How far do we need to go under Splitting, for introducing fresh variables starting from any given variable of $\mathcal{P}$?

Answer:
the splitting depth (sp-depth) of that variable, defined below.

Technical Notions - Step 1

First a relation:
$U \sim V$ is the finest equivalence relation on $\mathcal{X} = \mathcal{X}_{\mathcal{P}}$ such that:

- if $U = V \in \mathcal{P}$ then $U \sim V$;
- if $U = enc(V, T) \in \mathcal{P}$ or $V = enc(U, T) \in \mathcal{P}$, then $U \sim V$;
- if two pairings of the form $W = U.X, \ W' = V.X'$ are in $\mathcal{P}$, with $W \sim W'$, then $U \sim V$ and $X \sim X'$.

For any $Z \in \mathcal{X} = \mathcal{X}_{\mathcal{P}}$, define:
sp-depth of Z = maximum number of p_1- or p_2- steps from Z to all possible $X \in \mathcal{X}$, along the loop-free chains formed of $\sim$- or p_1/p_2-steps from Z to X.

Technical Notions - Step 1

We then observe:

Suffices to look for *discriminating solutions* for $\mathcal{P}$; that is to say:

Ground solutions in HE-normal form, assigning
distinct values to distinct key variables of $\mathcal{P}$.

Reason: A non-discriminating solution for $\mathcal{P}$ is a discriminating solution for a *Variant* of $\mathcal{P}$ derivable under a suitable inference ('*Equate some Keys*')

Technical Notions - Step 1

Necessary condition **SNF**, for $\mathcal{P}$ to admit a discriminating solution:

For any directed loop from a node Z to itself on the graph G, each arc of which is labeled by a homomorhism in $\mathcal{H}$, the word $\alpha \in \mathcal{H}^*$ labeling the loop must simplify to ϵ under the rules:

$$h_T \overline{h}_T \to \epsilon, \quad \overline{h}_T h_T \to \epsilon, \qquad T \in \mathcal{K}_{\mathcal{P}}$$

Technical Notions - Step 1

Necessary condition **SNF**, for $\mathcal{P}$ to admit a discriminating solution:

For any directed loop from a node Z to itself on the graph G, each arc of which is labeled by a homomorhism in $\mathcal{H}$, the word $\alpha \in \mathcal{H}^*$ labeling the loop must simplify to ϵ under the rules:

$$h_T\overline{h}_T \to \epsilon, \quad \overline{h}_T h_T \to \epsilon, \qquad T \in \mathcal{K}_{\mathcal{P}}$$

Reason:
If σ is such a solution, and $Z \rightarrowtail^{\alpha} Z$ a non-trivial loop on G formed of $h/\overline{h}$-arcs, the ground term $\sigma(\alpha)(\sigma Z)$ must normalize to $\sigma(Z)$; that can be done only by the two rewrite rules in HE:

$$dec(enc(x, y), y) \to x, \qquad enc(dec(x, y), y) \to x$$

So we add a Failure Inference rule if SNF not satisfied.

Technical Notions - Step 1

Necessary condition **SNF**, for $\mathcal{P}$ to admit a discriminating solution:

> For any directed loop from a node Z to itself on the graph G, each arc of which is labeled by a homomorhism in $\mathcal{H}$, the word $\alpha \in \mathcal{H}^*$ labeling the loop must simplify to ϵ under the rules:
>
> $$h_T \overline{h}_T \to \epsilon, \quad \overline{h}_T h_T \to \epsilon, \qquad T \in \mathcal{K}_{\mathcal{P}}$$

Reason:
If σ is such a solution, and $Z \rightarrowtail^{\alpha} Z$ a non-trivial loop on G formed of $h/\overline{h}$-arcs, the ground term $\sigma(\alpha)(\sigma Z)$ must normalize to $\sigma(Z)$; that can be done only by the two rewrite rules in HE:

$$dec(enc(x, y), y) \to x, \qquad enc(dec(x, y), y) \to x$$

So we add a Failure Inference rule if SNF not satisfied.

Consequence:

> Between any two nodes on the graph of a simple problem, **unique** directed loop-free path

Method-Step 1: Results

Proposition 1. On any problem $\mathcal{P}$ given in standard form, the Inference procedure terminates. In case of non-Failure, it returns an admissible problem $\mathcal{P}_1$ equivalent to $\mathcal{P}$.

Remark. Number of equations in $\mathcal{P}_1$ can be exponential wrt that in $\mathcal{P}$.
A typical example:

$$
\begin{array}{lll}
X_1 = enc(X_2, U_1) & X_{11} = enc(X_{12}, U_2) & X_{111} = enc(X_{112}, U_3) \\
X_1 = X_{11}.X_{12} & X_{11} = X_{111}.X_{112} & X_{111} = X_{1111}.X_{1112}
\end{array}
$$

Method-Step 1: Results

From now on our problems $\mathcal{P}$ are assumed admissible.

Method-Step 1: Results

From now on our problems $\mathcal{P}$ are assumed admissible.

As observed earlier:
If $Z \rightarrowtail^{\alpha} X$ is a directed path on G, labeled with word $\alpha \in \mathcal{H}^*$,
then Z evaluable by applying α to the evaluation of X,
and/or X evaluable by applying $\bar{\alpha}$ to the evaluation of Z.

Makes sense only if: $\tilde{h}_Z \notin \alpha$ or $\tilde{h}_X \notin \bar{\alpha}$, where $\tilde{h}$ stands for h or $\overline{h}$.

Method-Step 1: Results

From now on our problems $\mathcal{P}$ are assumed admissible.

As observed earlier:
If $Z \rightarrowtail^{\alpha} X$ is a directed path on G, labeled with word $\alpha \in \mathcal{H}^*$,
then Z evaluable by applying α to the evaluation of X,
and/or X evaluable by applying $\bar{\alpha}$ to the evaluation of Z.

Makes sense only if: $\tilde{h}_Z \notin \alpha$ or $\tilde{h}_X \notin \bar{\alpha}$, where $\tilde{h}$ stands for h or $\overline{h}$.

So notion of *No-Key-Dependency-Cycle* (NKDC):

> $Z \succ_k X$ iff $Z \neq X$, there is a directed path from Z to X on $G_{\mathcal{P}}$, an arc of which is labeled with $h_{X'}$ or $\overline{h}_{X'}$, with $X' = X$ or $X' \gtrdot X$.
>
> (**NKDC**) The graph $G = G_{\mathcal{P}}$ does not contain any node X such that $X \succ_k^+ X$, where $\succ_k^+$ = transitive closure of $\succ_k$.

Method-Step 1: Results

From now on our problems $\mathcal{P}$ are assumed admissible.

As observed earlier:
If $Z \rightarrowtail^{\alpha} X$ is a directed path on G, labeled with word $\alpha \in \mathcal{H}^*$,
then Z evaluable by applying α to the evaluation of X,
and/or X evaluable by applying $\bar{\alpha}$ to the evaluation of Z.

Makes sense only if: $\tilde{h}_Z \notin \alpha$ or $\tilde{h}_X \notin \bar{\alpha}$, where $\tilde{h}$ stands for h or $\overline{h}$.

So notion of *No-Key-Dependency-Cycle* (NKDC):

> $Z \succ_k X$ iff $Z \neq X$, there is a directed path from Z to X on $G_{\mathcal{P}}$,
> an arc of which is labeled with $h_{X'}$ or $\overline{h}_{X'}$, with $X' = X$ or $X' \gtrdot X$.
>
> (**NKDC**) The graph $G = G_{\mathcal{P}}$ does not contain any node X
> such that $X \succ_k^+ X$, where $\succ_k^+$ = transitive closure of $\succ_k$.

Proposition 2. An admissible $\mathcal{P}$ has a discriminating solution
if and only if its graph G satisfies NKDC.

Step 2: Solving a Simple problem

From now, we assume our problem $\mathcal{P}$ to be simple, and that its graph G satisfies NKDC.

Step 2: Solving a Simple problem

From now, we assume our problem $\mathcal{P}$ to be simple,
and that its graph G satisfies NKDC.

To solve such a $\mathcal{P}$:

(i) Choose a *base-node* V on each connected component Γ of G:
a node V on Γ that is *minimal* for the relation $\succ_k^+$.

(ii) For any node Z on Γ, assign the value $\alpha(V)$ where $\alpha \in \mathcal{H}^*$ is the word that labels **the** path from Z to V (V is uninstantiated, unless necessary).

Step 2: Solving a Simple problem

From now, we assume our problem $\mathcal{P}$ to be simple,
and that its graph G satisfies NKDC.

To solve such a $\mathcal{P}$:
(i) Choose a *base-node* V on each connected component Γ of G:
a node V on Γ that is *minimal* for the relation $\succ_k^+$.
(ii) For any node Z on Γ, assign the value $\alpha(V)$ where $\alpha \in \mathcal{H}^*$ is the word
that labels **the** path from Z to V (V is uninstantiated, unless necessary).

Example 2. Following problem $\mathcal{P}'$ is simple, its graph is connected:

$$X = enc(U, V),\ U = enc(V, T),\ V = enc(Y, U)$$

$$\begin{tikzcd} X \arrow[r, shift right, "h_V"'] & U \arrow[l, shift right, "\overline{h}_V"'] \arrow[r, shift right, "h_T"'] & V \arrow[l, shift right, "\overline{h}_T"'] \arrow[r, shift right, "h_U"'] & Y \arrow[l, shift right, "\overline{h}_U"'] \end{tikzcd}$$

Key-dependencies: $X \succ_k V$ and $Y \succ_k U$, so NKDC holds.
Both U and V are minimal for $\succ_k^+$. Choosing U as base-node gives the following discriminating solution for $\mathcal{P}'$:

$$V = \overline{h}_T(U),\ Y = \overline{h}_U(V) = \overline{h}_U\overline{h}_T(U),\ X = h_V(U).$$

Step 3: Solving an Admissible problem

So we get: If $\mathcal{P}$ is admissible, and Graph G of $\mathcal{P}$ satisfies NKDC, let $\mathcal{P}'$ = kernel of $\mathcal{P}$, G' its graph. Then we know how to solve $\mathcal{P}'$.

Step 3: Solving an Admissible problem

So we get: If $\mathcal{P}$ is admissible, and Graph G of $\mathcal{P}$ satisfies NKDC,
let $\mathcal{P}'$ = kernel of $\mathcal{P}$, G' its graph. Then we know how to solve $\mathcal{P}'$.

On any connected compoent Γ of G, define an *end-node* for $\mathcal{P}$ on Γ as any node $X \in \Gamma$ such that:

- there is an incoming path at X *only formed of* p_1/p_2*- arcs*;
- there is no outgoing arc from X.

Note: Path from any node Z on G to any given end-node is **unique**.

Step 3: Solving an Admissible problem

So we get: If $\mathcal{P}$ is admissible, and Graph G of $\mathcal{P}$ satisfies NKDC,
let $\mathcal{P}'$ = kernel of $\mathcal{P}$, G' its graph. Then we know how to solve $\mathcal{P}'$.

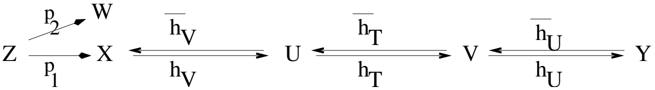
On any connected compoent Γ of G, define an *end-node* for $\mathcal{P}$ on Γ as any node $X \in \Gamma$ such that:

- there is an incoming path at X *only formed of* p_1/p_2 *- arcs*;
- there is no outgoing arc from X.

Note: Path from any node Z on G to any given end-node is **unique**.

Example 2b: Problem $\mathcal{P}$ below is admissible, its graph G is connected:

$$Z = X.W,\ X = enc(U, V),\ U = enc(V, T),\ V = enc(Y, U)$$



W is the only end-node here.
(In general: there may be many end-nodes, or none at all)

Solving an Admissible problem (contd.)

To solve an admissible $\mathcal{P}$, its graph G satisfying NKDC,
$\mathcal{P}'$ = kernel of $\mathcal{P}$, $G' = G_{\mathcal{P}'}$ seen as subgraph of G:

- On each connected component Γ of G, choose **one** base-node, and **all** the end-nodes, if any.
- Choose a solution σ' for $\mathcal{P}'$, that is minimal in the sense: does not instantiate any node of G not on G'
- For any $Z \in \Gamma$, assign the value obtained by 'propagating' the value assigned by σ' to the chosen base-node on Γ and the values assigned to the end-nodes on Γ, if any.

Propagation = use the homomorphisms and/or projections labeling the arcs along (uniquely determined) paths.

Example 2b - contd.

Example 2b (contd): For problem $\mathcal{P}$ of Example 2b,
kernel = problem $\mathcal{P}'$ of Example 2.

We got the following solution for $\mathcal{P}'$:

$$V = \overline{h}_T(U),\ Y = \overline{h}_U(V) = \overline{h}_U\overline{h}_T(U),\ X = h_V(U).$$

Propagation assigns the value $h_V(U).W$ to variable Z
(W a priori unistantiated, unless gets a specific value, e.g. with $W = a$).

Solving HE-Unifn problems: Results

Proposition 3.

(a) Solving an HE-Unifn problem given in standard form, by the above method, is in NEXPTIME wrt its number of equations.

(b) Solving simple HE-Unifn problems is NP-hard.

Solving HE-Unifn problems: Results

Proposition 3.

(a) Solving an HE-Unifn problem given in standard form, by the above method, is in NEXPTIME wrt its number of equations.

(b) Solving simple HE-Unifn problems is NP-hard.

(a) The NEXPTIME upper bound:

- Transforming $\mathcal{P}$ to an admissible $\mathcal{P}_1$ is in NEXPTIME
- Solving the kernel $\mathcal{P}_1'$ of $\mathcal{P}_1$ is in NP
  (amounts to checking for NKDC on its graph)
- Solving subsequently $\mathcal{P}_1$ is in P.

Solving HE-Unifn problems: Results

Proposition 3.

(a) Solving an HE-Unifn problem given in standard form, by the above method, is in NEXPTIME wrt its number of equations.

(b) Solving simple HE-Unifn problems is NP-hard.

(a) The NEXPTIME upper bound:

- Transforming $\mathcal{P}$ to an admissible $\mathcal{P}_1$ is in NEXPTIME
- Solving the kernel $\mathcal{P}_1'$ of $\mathcal{P}_1$ is in NP
  (amounts to checking for NKDC on its graph)
- Solving subsequently $\mathcal{P}_1$ is in P.

(b) NP-lower bound via reduction from the following *Monotone 1-in-3 SAT* problem - known to be NP-complete:

Solving HE-Unifn problems: Results

The Monotone 1-in-3 SAT problem:

> Given a propositional formula without negation, in CNF over 3 variables, check for satisfiability under the assumption that *exactly* one literal in each clause evaluates to true.

Solving HE-Unifn problems: Results

The Monotone 1-in-3 SAT problem:

> Given a propositional formula without negation, in CNF over 3 variables, check for satisfiability under the assumption that *exactly* one literal in each clause evaluates to true.

Let $\mathcal{P}$ = simple problem derived from the following HE-Unifn problem, in 3 variables x_1, x_2, x_3, and ground constants a, b, c:

$$dec(enc(dec(enc(dec(enc(a,b),x_1),b),x_2),b),x_3) = dec(enc(a,b),c).$$

Solving this $\mathcal{P}$:
Exactly one of the three variables x_1, x_2, x_3 is assigned the value c.

Concluding Remarks

Other possible specs for Hom. Encryption

1. HE$_1$: with 'Pairings'. Decrypt = "Encrypt with Inverse key"

$$\begin{array}{rcl} p_1(x.y) & \rightarrow & x \\ p_2(x.y) & \rightarrow & y \end{array} \qquad \begin{array}{rcl} enc(x.y,z) & \rightarrow & enc(x,z).enc(y,z) \\ enc(enc(x,f(y)),g(y)) & \rightarrow & x \\ enc(enc(x,g(y)),f(y)) & \rightarrow & x \end{array}$$

Method works almost unchanged for HE$_1$.

Concluding Remarks

Other possible specs for Hom. Encryption

1. HE$_1$: with 'Pairings'. Decrypt = "Encrypt with Inverse key"

$$
\begin{array}{rcl}
p_1(x.y) & \rightarrow & x \\
p_2(x.y) & \rightarrow & y
\end{array}
\qquad
\begin{array}{rcl}
enc(x.y, z) & \rightarrow & enc(x, z).enc(y, z) \\
enc(enc(x, f(y)), g(y)) & \rightarrow & x \\
enc(enc(x, g(y)), f(y)) & \rightarrow & x
\end{array}
$$

Method works almost unchanged for HE$_1$.

2. HE$_2$ = Drop the p_1, p_2 rules of HE$_1$:

$$
\begin{array}{rcl}
enc(x.y, z) & \rightarrow & enc(x, z).enc(y, z) \\
enc(enc(x, f(y)), g(y)) & \rightarrow & x \\
enc(enc(x, g(y)), f(y)) & \rightarrow & x
\end{array}
$$

Models RSA, if '.' is integer multiplication (mod suitable integer N),

Encrypt = exponentiation mod N with 'public key'

Decrypt = exponentiation mod N with 'private key'

Method unchanged for simple problems. Inferences on Pairings modified appropriately, for the "combination reasoning" to go through.

Concluding Remarks - contd.

3. HE$_3$: with 'Pairings', but *dec only left-inverse for enc*:

$$
\begin{array}{rclrcl}
p_1(x.y) & \rightarrow & x & enc(x.y, z) & \rightarrow & enc(x, z).enc(y, z) \\
p_2(x.y) & \rightarrow & y & dec(enc(x, y), y) & \rightarrow & x
\end{array}
$$

(HE$_3$ *can be made convergent by adding a meta- or schematized rule.*)

Concluding Remarks - contd.

3. HE$_3$: with 'Pairings', but *dec only left-inverse for enc*:

$$
\begin{array}{rclrcl}
p_1(x.y) & \rightarrow & x & enc(x.y, z) & \rightarrow & enc(x, z).enc(y, z) \\
p_2(x.y) & \rightarrow & y & dec(enc(x, y), y) & \rightarrow & x
\end{array}
$$

(HE$_3$ *can be made convergent by adding a meta- or schematized rule.*)

Method above doesn't work for HE$_3$. But *Active Deduction* modulo HE$_3$ can be shown to be decidable, via an Inference procedure for solving Cap Constraints, based on 'Cap Unification' ("no combination" here..!!); cf. UNIF-2009.

Concluding Remarks - contd.

3. HE$_3$: with 'Pairings', but *dec only left-inverse for enc*:

$$\begin{array}{lcl} p_1(x.y) & \rightarrow & x \\ p_2(x.y) & \rightarrow & y \end{array} \qquad \begin{array}{rcl} enc(x.y, z) & \rightarrow & enc(x, z).enc(y, z) \\ dec(enc(x, y), y) & \rightarrow & x \end{array}$$

(HE$_3$ *can be made convergent by adding a meta- or schematized rule.*)

Method above doesn't work for HE$_3$. But *Active Deduction* modulo HE$_3$ can be shown to be decidable, via an Inference procedure for solving Cap Constraints, based on 'Cap Unification' ("no combination" here..!!); cf. UNIF-2009.

So, unification modulo HE$_3$ decidable too (implicitly).

Concluding Remarks - contd.

As observed earlier, the Hom. Encryption specs considered above model crypto-protocols using ECB block-ciphering.

But ECB known to be vulnerable, so encryption might use ‘CBC-based’ paddings of message blocks.

Concluding Remarks - contd.

As observed earlier, the Hom. Encryption specs considered above model crypto-protocols using ECB block-ciphering.

But ECB known to be vulnerable, so encryption might use 'CBC-based' paddings of message blocks.

The convergent AC-TRS below models a CBC-based padding of *enc* with XOR (here '$+$' stands for XOR, and is AC):

$$
\begin{array}{ll}
x + 0 \to x, & x + x \to 0 \\
p_1(cons(x, y)) \to x, & p_2(cons(x, y)) \to y \\
dec(enc(x, y), y) \to x & \\
\multicolumn{2}{l}{cbc(cons(x, y), z, w) \to cons(\, enc(z + x, w), cbc(y, enc(z + x, w), w)\,)} \\
cbc(nil, z, k) \to nil &
\end{array}
$$

Concluding Remarks - contd.

As observed earlier, the Hom. Encryption specs considered above model crypto-protocols using ECB block-ciphering.

But ECB known to be vulnerable, so encryption might use 'CBC-based' paddings of message blocks.

The convergent AC-TRS below models a CBC-based padding of *enc* with XOR (here '$+$' stands for XOR, and is AC):

$$
\begin{array}{ll}
x + 0 \to x, & x + x \to 0 \\
p_1(cons(x, y)) \to x, & p_2(cons(x, y)) \to y \\
dec(enc(x, y), y) \to x & \\
\multicolumn{2}{l}{cbc(cons(x, y), z, w) \to cons(\, enc(z + x, w), cbc(y, enc(z + x, w), w)\,)} \\
cbc(nil, z, k) \to nil &
\end{array}
$$

Active deduction modulo this TRS: ongoing work...

Passive deduction modulo this TRS can be shown to be decidable (However passive deduction might be 'unrelated' to unification!!).