Implementing Cryptographic Primitives in the Symbolic Model

Peeter Laud
Cybernetica AS & Tartu University
http://www.cs.ut.ee/~peeter_l

EST-LAT Theory Days, Rakari, 01.10.2010

Applied pi calculus

Processes

$$
\begin{array}{rcll}
P, Q & ::= & 0 & \text{do nothing} \\
& | & \overline{c}\langle M \rangle.P & \text{Send } M \text{ on } c\text{, then do } P \\
& | & c(x).P & \text{Receive a message on } c\text{, bind it to } x \text{ in } P \\
& | & \nu n.P & \text{“generate a new” name } n\text{, then do } P \\
& | & P \mid Q & \text{Run } P \text{ and } Q \text{ in parallel} \\
& | & !P & \text{same as } P \mid !P \\
& | & [M = N]\ ?\ P : Q &
\end{array}
$$

Messages (terms)

$$
\begin{array}{rcll}
M, N & ::= & n & \text{name} \\
& | & x & \text{variable} \\
& | & f(M_1, \ldots, M_k) & \textbf{function symbol } \text{application}
\end{array}
$$

Applied pi calculus

Processes

$$P, Q \quad ::= \quad 0$$ do nothing

$| \quad \overline{c}\langle M \rangle.P$ Send *M* c

$| \quad c(x).P$ Receive a message on *c*, bind it to *x* in *P*

$| \quad \nu n.P$ "generate a new" name *n*, then do *P*

$| \quad P \mid Q$ Run *P* and *Q* in parallel

$| \quad !P$ same as *P* | !*P*

$| \quad [M = N]\ ?\ P : Q$

Communication:

$$c\langle M \rangle.P \mid c(x).Q \to P \mid Q[x \leftarrow M]$$

Messages (terms)

$$M, N \quad ::= \quad n$$ name

$| \quad x$ variable

$| \quad f(M_1, \ldots, M_k)$ **function symbol** application

Applied pi calculus

Processes

$$
\begin{array}{rcl}
P, Q & ::= & 0 \\
& | & \overline{c}\langle M \rangle.P \\
& | & c(x).P \\
& | & \nu n.P \\
& | & P \mid Q \\
& | & !P \\
& | & [M = N]\,?\,P : Q
\end{array}
$$

do nothing

Send M c

Receive a message on c, bind

“generate a new” name n, the

Run P and Q in parallel

same as $P \mid !P$

Communication:

$$c\langle M \rangle.P \mid c(x).Q \rightarrow P \mid Q[x \leftarrow M]$$

$f \in \Sigma$ — a **signature**

signature – a finite set of function symbols with associated **arities**

Messages (terms)

$$
\begin{array}{rcl}
M, N & ::= & n \\
& | & x \\
& | & f(M_1, \ldots, M_k)
\end{array}
$$

name

variable

function symbol application

Applied pi calculus

Processes

$$P, Q \quad ::= \quad 0$$

do nothing

$$\mid \quad \overline{c}\langle M \rangle.P$$

Send M c

$$\mid \quad c(x).P$$

Receive a message on c, binc

$$\mid \quad \nu n.P$$

$$\mid \quad P \mid Q$$

$$\mid \quad !P$$

$$\mid \quad [M =$$

Communication:

$$c\langle M \rangle.P \mid c(x).Q \rightarrow P \mid Q[x \leftarrow M]$$

$f \in \Sigma$ — a **signature**

re – a finite set of

n symbols with

ated **arities**

Associated **equational theory**

E – a set of pairs of terms

$(M, N) \in E$ implies that we judge $M\sigma = N\sigma$ for all substitutions σ that ground M and N

Messages (terms)

$$M, N \quad ::= \quad n$$

$$\mid \quad x$$

variable

$$\mid \quad f(M_1, \ldots, M_k)$$

function symbol application

Applied pi calculus

Processes

$$P, Q \quad ::= \quad 0$$

do nothing

$$\mid \quad \overline{c}\langle M\rangle.P$$

Send *M* c

$$\mid \quad c(x).P$$

Receive a message on *c*, binc

$$\mid \quad \nu n.P$$

$$\mid \quad P \mid Q$$

$$\mid \quad !P$$

$$\mid \quad [M =$$

Messages (terms)

$$M, N \quad ::= \quad n$$

$$\mid \quad x$$

variable

$$\mid \quad f(M_1, \ldots, M_k)$$

function symbol application

Communication:

$$c\langle M\rangle.P \mid c(x).Q \rightarrow P \mid Q[x \leftarrow M]$$

$f \in \Sigma$ — a **signature**

re – a finite set of
n symbols with
ated **arities**

Associated **equational theory**

E – a set of pairs of terms

$(M, N) \in E$ implies that we judge $M\sigma = N\sigma$ for all substitutions *σ* that ground *M* and *N*

For example

$$\mathsf{fst}((x, y)) = x$$

$$\mathsf{snd}((x, y)) = y$$

Cryptography with applied pi calculus

- Signature – cryptographic and other operations
- Equational theory captures cryptographic identities
  - **lack of equations captures security**
- E.g. symmetric *randomized* encryption:
  - enc/3, dec/2 (need more later)
  - $dec(k,enc(r,k,x)) = x$
- A very useful abstraction of the computational model
  - (sometimes unsound)

Primitives

- Cryptography in computational model is all about building primitives
  - Start from base primitives with certain security properties
    - one-way functions, trapdoor one-way functions
  - Combine them into more complex primitives
    - reduce their security to security of simpler primitives
  - Use them to build your system
- In symbolic model, the set of primitives is given by the signature Σ

Our motivation

- We have obtained certain results for a certain set of primitives Σ
- We want to generalize those results to a larger set of primitives Σ'
- This would be straightforward if we could implement the primitives in Σ' using the primitives in Σ
- What does "implement" mean?

Motivating example

Consider the following primitives

tuples $(,\ldots,)/n$ $\pi_i^n/1$

$$\pi_i^n((x_1,\ldots,x_n)) = x_i$$

hashing $H/1$

H(x_1,...,x_k) is syntactic sugar for H((x_1,...,x_k))

XOR $\oplus/2$ $0/0$

$$x \oplus y = y \oplus x$$

$$(x \oplus y) \oplus z = x \oplus (y \oplus z)$$

$$x \oplus x = 0 \qquad x \oplus 0 = x$$

how to implement

S.R.Enc $enc/3$ $dec/2$

$$dec(k, enc(r,k,x)) = x$$

with them?

What is an implementation?

for each symbol *f* of arity *k* in the primitive,
a term f^{i} with free variables $x_1,\dots,x_k$

- Must be compatible with the equational theory
- Induces a mapping *tr* on terms
  - (second-order substitution)
  - Replaces each occurrence of *f* with f^{i}
- *tr* is straightforwardly extended to processes
  - Replace each *M* with *tr*(*M*)
- Secure implementation: no *P* can be told apart from *tr*(*P*)

Observational equivalence

is the largest relation ≈ on closed processes, such that *P*≈*Q* implies

Alone, *P* and *Q* look the same

- *P* and *Q* have the same barbs
  - *P* has barb *c* if *P* can evolve to output on channel *c*
- If *P* can evolve to *P′* then *Q* can evolve to *Q′*, such that *P′*≈*Q′*
  - and *vice versa*
- For any closing evaluation context *C*, *C*[*P*]≈*C*[*Q*]
  - evaluation context is a process with a hole "in the front"
    - not preceded by I/O or conditionals

Secure implementation: *P*≈tr(*P*) for all *P* ???

Implementing Symm. Rand. Enc.

- Available operations: hashing and XOR
- Hashing looks like (pseudo)random function
  - $H(k,x)$ – keyed pseudorandom function
- From comp. model: to encrypt x with key k,
  - generate a random r
  - use H to expand it to random bit-string of length $|x|$
  - XOR it with x
- $(r, H(k,r) \oplus x)$ — might this be $enc^{i}_{[r,k,x]}$?
- $dec^{i}_{[k,y]}$ would then be $H(k, \pi_1^2(y)) \oplus \pi_2^2(y)$

Obs. equiv. is unsuitable

- Consider the following process *P*
  - construct a ciphertext
  - check whether it is a pair
  - depending on outcome, do observably different things
- *P*≈tr(*P*) cannot hold for such *P*
  - Note: the test could be performed by either *P* or the context *C*
- Our solution: *P* and *C* do not use the function symbols used for implementation
  - these symbols are "implementation details"

M is a pair if
$(\pi_1^2(M), \pi_2^2(M)) = M$

Obs. equiv. is unsuitable

- Consider the following process P
  - co
  - cl
  - d
    th

if
$= M$

P and *C* do not use pairings???

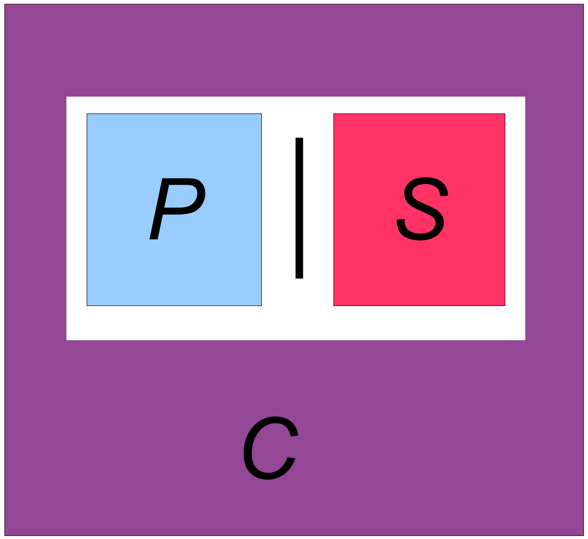
We assume, there are separate, "tagged" versions:

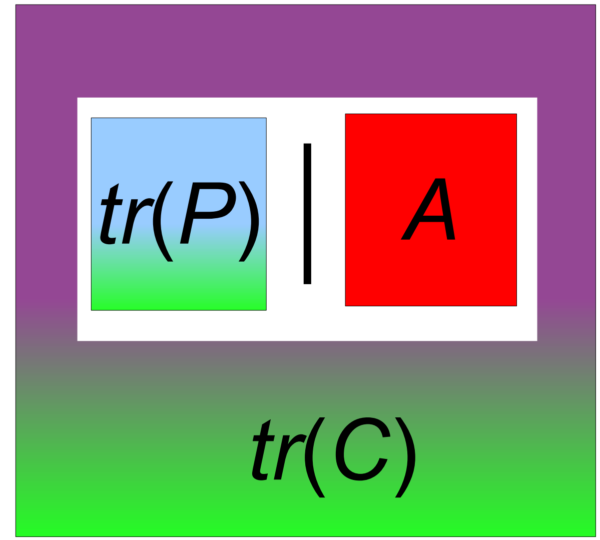
$$\bar{(}x,y\bar{)},\ \bar{\pi}_1(x),\ \bar{\pi}_2(x),\ \bar{H}(x)$$

and these are used only in the implementation

- *P*≈tr(*P*) cannot hold for such *P*
  - Note: the test could be performed by either *P* or the context *C*
- Our solution: *P* and *C* do not use the function symbols used for implementation
  - these symbols are "implementation details"

Secure implementation: intuition

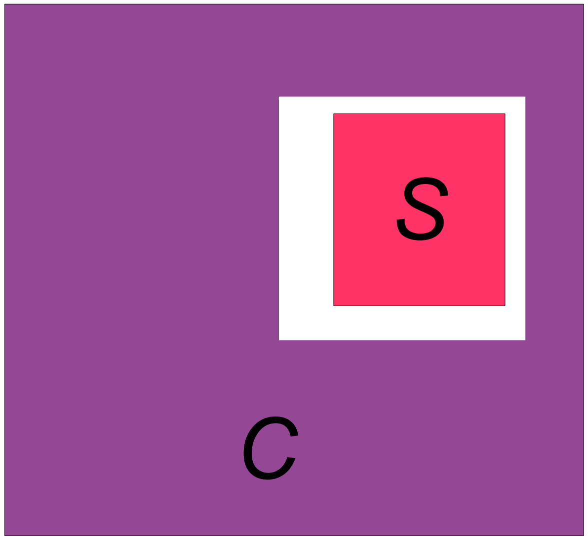


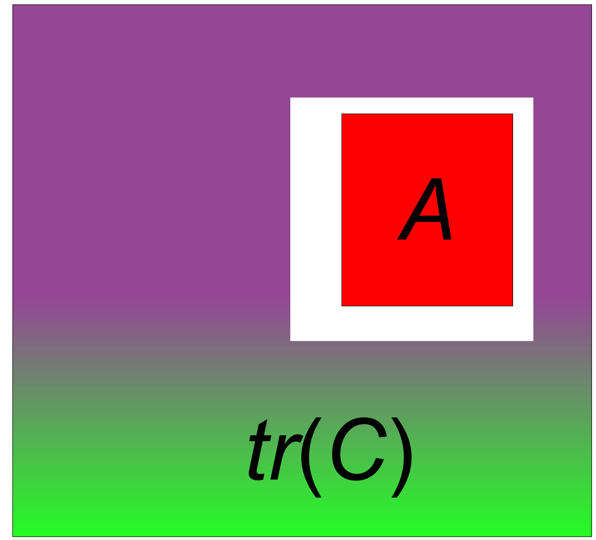


A can use tagged operations. *P*,*S*,*C* cannot

$\forall P \forall A \exists S \forall C$: the two processes have the same barbs

Simplification





A can use tagged operations. *S*,*C* cannot

$\forall A \exists S \forall C$: the two processes have the same barbs

Obs. equiv. modulo implementation

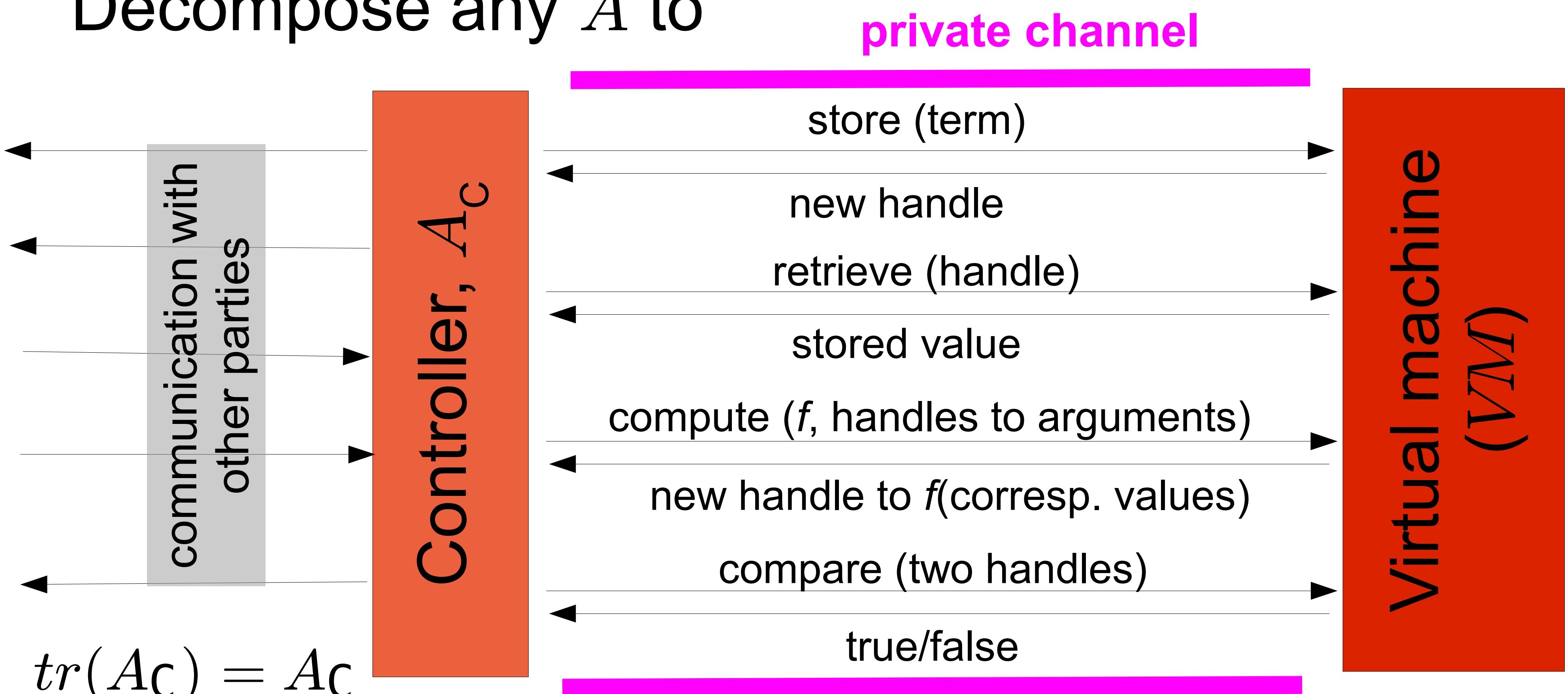
is the largest relation $\approx_{tr}$ on closed processes, such that $P \approx_{tr} Q$ implies

- P and Q have the same barbs
- If P can evolve to P' then Q can evolve to Q', such that $P' \approx_{tr} Q'$
  - and *vice versa*
- For any closing evaluation context C not using tagged symbols, $C[P] \approx_{tr} tr(C)[Q]$

Secure implementation: $\forall A \exists S : S \approx_{tr} A$
where S does not use tagged symbols

Proving security of implementation

Decompose any A to



inde-
pendent
of A

Find S, such that $S \approx_{tr} VM$

Then $\nu c(A_C \mid S) \approx_{tr} \nu c(A_C \mid VM) \approx A$

Back to encryption...

tuples $(,\ldots,)/n$ $\quad \pi_i^n/1$

$$\pi_i^n((x_1,\ldots,x_n)) = x_i$$

hashing $H/1$

XOR $\oplus/2$ $\quad 0/0$

$$x \oplus y = y \oplus x$$

$$(x \oplus y) \oplus z = x \oplus (y \oplus z)$$

$$x \oplus x = 0 \qquad x \oplus 0 = x$$

how to implement

S.R.Enc $enc/3$ $\quad dec/2$

$$dec(k, enc(r,k,x)) = x$$

with them?

Does

$$\left\{\begin{array}{l} enc^{\mathtt{i}}_{[r,k,x]} = \bar{(}r, \bar{H}(k,r) \oplus x\bar{)} \\ dec^{\mathtt{i}}_{[k,y]} = \bar{H}(k, \bar{\pi}_1(y)) \oplus \bar{\pi}_2(y) \end{array}\right\}$$

work?

No, it does not work

A can transform $\overline{(}r, \bar{H}(k, r) \oplus x\overline{)}$, x'

to $\overline{(}r, \bar{H}(k, r) \oplus x \oplus x'\overline{)}$

No S can transform $enc(r, k, x)$, x'

to $enc(r, k, x \oplus x')$

Symbolic encryption also provides **integrity**

Integrity

- Message authentication codes (MACs) are used in the computational model to provide integrity in symmetric settings
- **Theorem** (comp. model): random function is a good MAC
- *H* models a random function

Integrity

- Message authentication codes (MACs) are used in the computational model to provide integrity in symmetric settings
- **Theorem** (comp. model): random function is a good MAC
- *H* models a random function
- How about: $enc^{\mathtt{i}}_{[r,k,x]} = \bar{(}r, \bar{H}(k,r) \oplus x, \bar{H}(k,x)\bar{)}$

$$dec^{\mathtt{i}}_{[k,y]} = \bar{H}(k, \bar{H}(k, \bar{\pi}_1(y)) \oplus \bar{\pi}_2(y)) \stackrel{?}{=} \bar{\pi}_3(y) \triangleright \bar{H}(k, \bar{\pi}_1(y)) \oplus \bar{\pi}_2(y)$$

where $\stackrel{?}{=} \triangleright$ is a ternary function symbol and

$$x \stackrel{?}{=} x \triangleright y = y$$

Still needs improvement

Given $\overline{(}r, \bar{H}(k, r) \oplus x, \bar{H}(k, x))\overline{)}$

$\overline{(}r', \bar{H}(k, r') \oplus x', \bar{H}(k, x'))\overline{)}$

A can tell whether x = x'

Given $enc(r, k, x)$ $enc(r', k, x')$

No S can tell whether x = x'

Still needs improvement

Given $\overline{(}r, \bar{H}(k,r) \oplus x, \bar{H}(k,x)\overline{)}$

$\overline{(}r', \bar{H}(k,r') \oplus x', \bar{H}(k,x')\overline{)}$

A can tell whether x = x'

Given $enc(r,k,x)$ $enc(r',k,x')$

No S can tell whether x = x'

Randomize the MAC

$$enc^{\mathtt{i}}_{[r,k,x]} = \overline{(}r, \bar{H}(k,r) \oplus x, \bar{H}(k,r,x)\overline{)}$$

Still needs improvement

This can be seen as the encrypt-then-MAC construction

It has good properties in the computational model

Is it secure?

Randomize the MAC

$$enc^{\mathtt{i}}_{[r,k,x]} = \overline{(}r, \bar{H}(k,r) \oplus x, \bar{H}(k,r,x)\overline{)}$$

Securing the implementation

Given $\bar{(}r, \bar{H}(k, r) \oplus x, \bar{H}(k, r, x))\bar{)}$

$\bar{(}r, \bar{H}(k, r) \oplus x', \bar{H}(k, r, x'))\bar{)}$

A can compute $x \oplus x'$

Given $enc(r, k, x)$ $enc(r, k, x')$

No S can compute $x \oplus x'$

Securing the implementation

Given $\overline{(}r, \bar{H}(k,r) \oplus x, \bar{H}(k,r,x))\overline{)}$

$\overline{(}r, \bar{H}(k,r) \oplus x', \bar{H}(k,r,x'))\overline{)}$

A can compute $x \oplus x'$

Given $enc(r,k,x)$ $enc(r,k,x')$

No S can compute $x \oplus x'$

Make the randomness depend on k and x

$$enc^{\mathtt{i}}_{[r,k,x]} = \overline{(}r, \bar{H}(k, \bar{H}(k,r,x)) \oplus x, \bar{H}(k,r,x))\overline{)}$$

$$dec^{\mathtt{i}}_{[k,y]} = \bar{H}(k, \bar{\pi}_1(y), \bar{H}(k, \bar{\pi}_3(y)) \oplus \bar{\pi}_2(y)) \stackrel{?}{=} \bar{\pi}_3(y) \triangleright \bar{H}(k, \bar{\pi}_3(y)) \oplus \bar{\pi}_2(y)$$

This is secure implementation

Simulating VM

- Simulator S must respond to storing, retrieving, comparing, computing queries
  - Including tagged operations
- Simulator may not use tagged operations
- Simulator must be indistinguishable from VM
- Stronger property:

At no time can one find terms *M* and *N* over VM's / S's database, such that
tr(*M*)[VM] = tr(*N*)[VM] XOR *M*[S] = *N*[S]

no tagged symbols in *M* and *N*

Tables of VM's and S's databases

hnd	v

hnd	v	v'

ct	snd	thd	k	pt

ciphertext
we've seen or created

second part

third part

correct key
(if any)

plaintext

The primitive needs more operations

- The implementation of enc reveals the used random coins
  - this is acceptable and natural
- The simulator needs to recognize ciphertexts
  - $\bar{(}\bar{\pi}_1(v), \bar{\pi}_2(v), \bar{\pi}_3(v)\bar{)}$ must equal v iff v is ciphertext

S.R.Enc

$$enc/3 \quad rnd/1$$
$$dec/2 \quad ct?/1$$
$$true/0$$

$$dec(k, enc(r, k, x)) = x$$
$$rnd(enc(r, k, x)) = r$$
$$ct?(enc(r, k, x)) = true$$

The primitive needs more operations

- The … used
  ran…
  - th…

$$rnd^{\mathtt{i}}_{[y]} = \bar{\pi}_1(y)$$

$$ct?^{\mathtt{i}}_{[y]} = \overline{(}\bar{\pi}_1(y), \bar{\pi}_2(y), \bar{\pi}_3(y)\overline{)} \stackrel{?}{=} y \triangleright true$$

- The simulator needs to rec… ciphertexts
  - $\overline{(}\bar{\pi}_1(v), \bar{\pi}_2(v), \bar{\pi}_3(v)\overline{)}$ must equal … v is ciphertext

S.R.Enc

$enc/3$ $rnd/1$
$dec/2$ $ct?/1$
$true/0$

$$dec(k, enc(r, k, x)) = x$$
$$rnd(enc(r, k, x)) = r$$
$$ct?(enc(r, k, x)) = true$$

Simulation

- Store, retrieve, compare, apply “normal” symbols – as VM
  - Whenever a ciphertext appears – update ct table
    - Check with $ct?$
  - Whenever a key/plaintext is learned – update table
    - k is correct key for y if $enc(rnd(y),k,dec(k,y))=y$
- Tagged operations – first look in the ct table
  - Also update the ct table as much as possible
- Repeat query – repeat answer

Simulation

- $\bar{H}(x,y,z)$: key, plaintext, randomness known – insert whole row into ct table
- $\bar{H}(x,y)$: if y is not a tagged hash of a triple, then this invocation cannot be part of a ciphertext
- tagged triple – **must** return a ciphertext
  - If no suitable row in ct table – return random c-text
- $\bar{\pi}_1$ is the same as rnd
- Other projections – see ct table
- If ct table lookup fails – generate random name

Conclusions

- “Implementation” changes the signatures
  - we've proposed a suitable equivalence in this case
  - ... and a proof method
  - ... and did an example
- Finding secure implementations is trickier than expected
  - Randomness is treated differently in symbolic and computational models
- The simulations might yield new, interesting proofs in the computational model