On the computational soundness of cryptographically masked flows

Peeter Laud

peeter.laud@ut.ee

http://www.ut.ee/~peeter_l

Tartu University & Cybernetica AS & Institute of Cybernetics
Motivation

- Usual non-interference too strong for programs with encryption.
- Cryptographic security definitions
  - use complex domains,
  - are notationally heavy.
- The definitions for computational non-interference suffer from the same problems.
- Could we abstract from these definitions? Is there some formalism, where
  - the domain and the definition of non-interference were more “traditional”,
  - NI for a program in this domain would mean computational NI for the “same” program in the real-world semantics?
Cryptographically masked flows


A proposal for the formalism that abstracts away complexity-theoretic details, but leaves (most of) everything else intact.

Encryption is modeled non-deterministically.

Possibilistic non-interference with extra leniency for encrypted values.

Does NI in this model imply computational NI? Are cryptographically masked flows computationally sound?

Acknowledgement: the above question was asked by David Sands during our Dagstuhl-event.
The programming language

In this talk: The WHILE-language with extra operations:
- key generation, encryption, decryption
- pairing, projection

In the [AHS06]-paper: more . . .
- Parallel processes with global variables and message channels
- Two encryption schemes (one for public values only)
Semantics

- Big-step SOS from a configuration to a set of final states.
- The state consists of
  - The memory — mapping from variables to values;
  - The “key-stream” — the values of keys generated in the future.
- All operations, except encryption, are deterministic.
Encryption Systems

Three algorithms:
- $\mathcal{K}$ — key generation, zero arguments, probabilistic;
- $\mathcal{E}$ — encryption, two arguments, probabilistic;
- $\mathcal{D}$ — decryption, two arguments, deterministic.

Correctness: $\mathcal{D}(k, \mathcal{E}(r; k, x)) = x$ for all
  - keys $k$ that can be output by $\mathcal{K}$;
  - possible random coins $r$ used by $\mathcal{E}$.

The random coins used by $\mathcal{E}$ are called the *initial vector*.

$\mathcal{D}$ may produce an error.
Semantics

- Big-step SOS from a configuration to a set of final states.
- The state consists of
  - The memory — mapping from variables to values;
  - The “key-stream” — the values of keys generated (by $\mathcal{K}$) in the future.
- All operations, except encryption, are deterministic.
- Encryption models the randomized encryption algorithms of the real world:
  - To encrypt $x$ with the key $k$, choose an initial vector $r$ and compute $E(r; k, x)$.
  - In reality, $r$ is chosen probabilistically, here it is modeled by non-deterministic choice.
Low-equivalence of memories

- Let the variables be partitioned to $\text{Var}_H$ and $\text{Var}_L$.
- Let the values be tagged with their types — key, encryption, pair, other (integer).
- $n \sim_L n$;
- $k \sim_L k$;
- $x_1 \sim_L y_1 \land x_2 \sim_L y_2 \Rightarrow (x_1, x_2) \sim_L (y_1, y_2)$;
- $E(r; k_1, x_1) \sim_L E(r; k_2, x_2)$ for all $x_1, x_2, k_1, k_2$.
- $S_1 \sim_L S_2$ if $S_1(x) \sim_L S_2(x)$ for all $x \in \text{Var}_L$. 
Possibilistic non-interference

Program $P$ is non-interfering if

- for all states $S_1, S_2$ and keystreams $G_1, G_2$, such that $S_1 \sim_L S_2$
- let $S_i = \{ S' \mid (S_i, G_i) \rightarrow (S', G') \}$ for $i \in \{1, 2\}$, then
- for all $S'_1 \in S_1$
- there must exist $S'_2 \in S_2$
- such that $S'_1 \sim_L S'_2$.

(and vice versa)
“Real-world” semantics

- Big step SOS — maps an initial configuration to a probability distribution over final states.
- Let us not consider non-termination.
- And assume that the program terminates in a reasonable number of steps.
- Initial state is distributed according to some $D$.
- The program $P$ is non-interferent if no algorithm $A$ using a reasonable amount of resources can guess $b$ from

\[
b \leftarrow_R \{0, 1\}, \quad S_0, S_1 \leftarrow D
\]

\[
S' \leftarrow \llbracket P \rrbracket (S_b)
\]

give $(S_0|\text{var}_H, S'|\text{var}_L)$ to $A$
Soundness theorem

If the program $P$ satisfies the following conditions:

- $\ldots$

and the encryption system satisfies the following conditions

- IND-KDM-CPA- and INT-PTXT-security

and $P$ satisfies possibilistic non-interference

then $P$ satisfies computational non-interference.

The conditions put on $P$ should be verifiable in the possibilistic model.

Otherwise we lose the modularity of the approach.
Condition: ciphertexts only from $\mathcal{E}$

- $\sim_L$’s relaxed treatment of ciphertexts must be restricted to values produced by the encryption operation.

- Otherwise, consider the following program:

$$
k := \text{newkey}; \quad p_1 := \text{enc}(k, s)$$

$$
r := \text{getIV}(p_1); \quad p_2 := \widetilde{\text{enc}}(r + 1; k, s)$$

- Initial state ($\{s \mapsto v_s\}, v_k :: G$) is mapped to

$$
\left\{ \{p_1 \mapsto \mathcal{E}(v_r; v_k, v_s), p_2 \mapsto \mathcal{E}(v_r + 1; v_k, v_s)\} \mid v_r \in \text{Coins} \right\}
$$

that does not depend (for $\sim_L$) on initial secrets.
Counter mode of using a block cipher

- A good encryption system.
- If we used it on the previous slide, then we could learn
  \[ v_{s1} \oplus v_{s2}, \ v_{s2} \oplus v_{s3}, \ v_{s3} \oplus v_{s4}, \ldots \]
Security of encryption systems

Let $O_0$ and $O_1$ be the following interactive machines:
- on initialization, generate $k \leftarrow \mathcal{K}(\cdot)$;
- on query $x \in \{0, 1\}^*$
  - $O_0$ returns $E(k, x)$,
  - $O_1$ returns $E(k, 0|x|)$.

Encryption system is IND-CPA-secure if no reasonably powerful adversary $A$ can guess $b$ from the interaction with $O_b$.

IND-CPA with multiple keys: $O_0$ and $O_1$
- on initialization generate $k_i \leftarrow \mathcal{K}(\cdot)$ for all $i \in \mathbb{N}$;
- on query $(i, x)$ use the key $k_i$ for $x$ as before.

IND-CPA with multiple keys is equivalent to IND-CPA.
More security considerations

- Encryption cycles are not excluded, hence we must use encryption systems secure in the presence of key dependent messages.

- Our definition of possibilistic NI also hides
  - the identities of keys,
  - the length of messages.
Let $O_0$ and $O_1$ be the following:

- **On initialization**
  - $O_0$ generates keys $k_i, i \in \mathbb{N}$;
  - $O_1$ generates the key $k$.

- **On input** $(i, e)$ where $e$ is an expression with free variables $k_j$ the machine $O_0$
  - evaluates $e$, letting $k_j$ refer to its keys,
  - encrypts the result with $k_i$ and returns it;
  - and the machine $O_1$ returns $E(k, 0^\text{const})$.

If no reasonably powerful adversary $A$ can guess $b$ from the interaction with $O_b$ then the encryption system is IND-CPA-secure, which-key concealing and length-concealing in the presence of key-dependent messages.
Condition: keys used only at $\mathcal{E}$ and $\mathcal{D}$...

- ...and vice versa.
- Consider the program

$$k_1 := \text{newkey}; \text{if } B(k_1) \text{ then } k_2 := k_1 \text{ else } k_2 := \text{newkey} \text{ fi}; \ldots$$

- Afterwards, $k_2$ is not distributed as coming from $\mathcal{K}$. 
What may be decrypted

- The possibilistic semantics only allows to decrypt legitimate ciphertexts.
- We may phrase this as a condition on the programs.
- Or we may require that the encryption system provides *integrity for plaintexts*:

  Let $\mathcal{O}$ be the following:
  - On initialization, it generates $k \leftarrow \mathcal{K}()$;
  - On query $x$, it returns $\mathcal{E}(k, x)$.

  No reasonably powerful adversary $\mathcal{A}$ interacting with $\mathcal{O}$ may be able to produce a ciphertext $c$, such that
  - $\mathcal{D}(k, c) = m$ (i.e. $\mathcal{D}$ does not fail);
  - $\mathcal{A}$ did not query $\mathcal{O}$ with $m$. 
Enforcing those conditions

- Give types to variables: the types \( \tau \) are

\[
\tau ::= \text{int} \mid \text{key} \mid \text{enc}(\tau) \mid (\tau, \tau)
\]

- We may want to compute with ciphertexts, hence we subtype \( \text{enc}(\tau) \leq \text{int} \).

- Types of operations:
  - arithmetic operations: \( \text{int}^k \rightarrow \text{int} \);
  - pairing: \( \tau_1 \times \tau_2 \rightarrow (\tau_1, \tau_2) \);
  - \( i \)-th projection: \( (\tau_1, \tau_2) \rightarrow \tau_i \);
  - key generation: \( 1 \rightarrow \text{key} \);
  - encryption: \( \text{key} \times \tau \rightarrow \text{enc}(\tau) \);
  - decryption: \( \text{key} \times \text{enc}(\tau) \rightarrow \tau \);
  - guards: \( \text{int} \).

- [AHS06] already has such a type system.
Removing decryptions

- Change the real-world program:
  - Give names to keys: replace each $k := \text{newkey}$ with
    $$k := \text{newkey}; k_{\text{name}} := c; c := c + 1$$
  - for each ciphertext record the key name and the plaintext in the auxiliary variables. Replace
    $y := \mathcal{E}(k, x)$ with
    $$y := \mathcal{E}(k, x); y_{\text{keyname}} := k_{\text{name}}; y_{\text{ptext}} := x$$
  - Replace the statements $x := \mathcal{D}(k, y)$ with
    $$\text{if } k_{\text{name}} = y_{\text{keyname}} \text{ then } x := y_{\text{ptext}} \text{ else } x := \bot \text{ fi}$$
  - The low-visible semantics does not change.
Encryption $\rightarrow$ random number generation

- Apply the definition of IND-KDM-CPA to the real-world program:
  - Replace each $E(k, y)$ with $E(k_0, 0)$.
  - $E(k_0, 0)$ generates random numbers according to a certain distribution.

- In the possibilistic NI, we also treat encryption as random number generation.
  - As only the initial vector matters.
Possib. secrecy $\not\Rightarrow$ probab. secrecy

Let $h$ be a number from 1 to 100. Consider the following program:

```
if $\text{rnd}([0, 1]) = 1$ then $l := h$ else $l := \text{rnd}([1, \ldots, 100])$
```

The possible values of $l$ do not depend on $h$.

But their distribution depends on $h$.

We can come up with similar examples in our language.

- Using $\&$ in place of $\text{rnd}$.
- Hence using ciphertexts in computations is questionable as well.

- Remove the subtyping $\text{enc}(\tau) \leq \text{int}$.
The conditions for the program

- The variables are typed, as specified before.
  \[ \tau ::= \text{int} \mid \text{key} \mid \text{enc}(\tau) \mid (\tau, \tau) \]
  (no subtyping)

- The operations respect those types.

- Failures to decrypt are visible in the possibilistic semantics.

- Our theorem holds now.
  - In a program point, two ciphertexts are either equal or independent.