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

- Aslan Askarov, Daniel Hedin, Andrei Sabelfeld. Cryptographically-Masked Flows. SAS 2006.
- 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 $\mathcal{E}(r;k,x)$.
  - In reality, r is chosen probabilistically, here it is modeled by non-deterministic choice.

Low-equivalence of memories

- Let the variables be partitioned to $\mathbf{Var}_{\mathrm{H}}$ and $\mathbf{Var}_{\mathrm{L}}$.
- Let the values be tagged with their types — key, encryption, pair, other (integer).
- $n \sim_{\mathrm{L}} n$;
- $k \sim_{\mathrm{L}} k$;
- $x_1 \sim_{\mathrm{L}} y_1 \wedge x_2 \sim_{\mathrm{L}} y_2 \Rightarrow (x_1, x_2) \sim_{\mathrm{L}} (y_1, y_2)$;
- $\mathcal{E}(r; k_1, x_1) \sim_{\mathrm{L}} \mathcal{E}(r; k_2, x_2)$ for all x_1, x_2, k_1, k_2.
- $S_1 \sim_{\mathrm{L}} S_2$ if $S_1(x) \sim_{\mathrm{L}} S_2(x)$ for all $x \in \mathbf{Var}_{\mathrm{L}}$.

Possibilistic non-interference

Program P is non-interfering if

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

(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 $\mathcal{A}$ using a reasonable amount of resources can guess b from

$$b \leftarrow_R \{0,1\},\ S_0, S_1 \leftarrow D$$
$$S' \leftarrow [\![P]\!](S_b)$$
$$\text{give } (S_0|_{\mathbf{Var}_{\mathrm{H}}}, S'|_{\mathbf{Var}_{\mathrm{L}}}) \text{ to } \mathcal{A}$$

Soundness theorem

- If the program P satisfies the following conditions:
  - ...
- 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 $\mathcal{E}$

- $\sim_{\mathrm{L}}$'s relaxed treatment of ciphertexts must be restricted to values produced by the encryption operation.
- Otherwise, consider the following program:

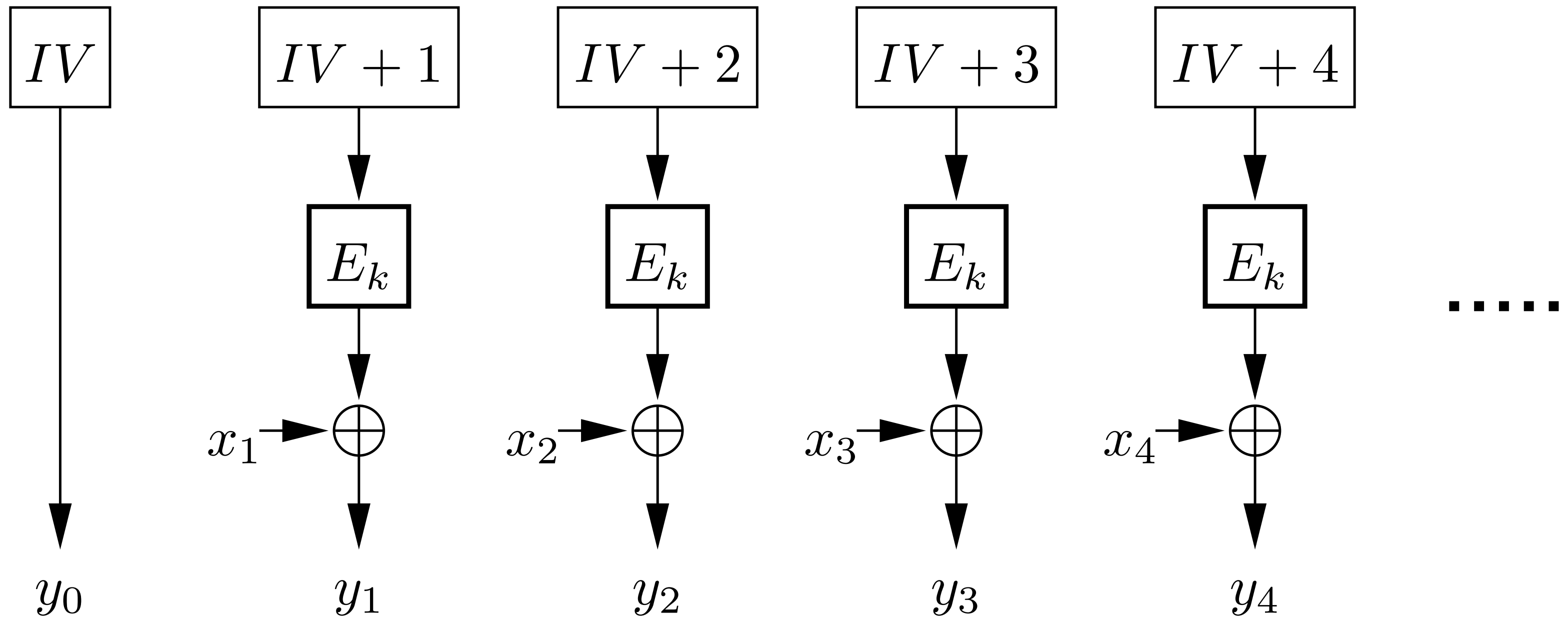
$$k := \mathsf{newkey}; p_1 := \mathsf{enc}(k, s)$$
$$r := \mathsf{getIV}(p_1); p_2 := \widetilde{\mathsf{enc}}(r + 1; k, s)$$

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

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

that does not depend (for $\sim_{\mathrm{L}}$) 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 $\mathcal{O}_0$ and $\mathcal{O}_1$ be the following interactive machines:
  - on initialization, generate $k \leftarrow \mathcal{K}()$;
  - on query $x \in \{0,1\}^*$
    - $\mathcal{O}_0$ returns $\mathcal{E}(k, x)$,
    - $\mathcal{O}_1$ returns $\mathcal{E}(k, 0^{|x|})$.
- Encryption system is IND-CPA-secure if no reasonably powerful adversary $\mathcal{A}$ can guess b from the interaction with $\mathcal{O}_b$.
- IND-CPA with multiple keys: $\mathcal{O}_0$ and $\mathcal{O}_1$
  - on initialization generate $k_i \leftarrow \mathcal{K}()$ 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.

IND-KDM-CPA

- Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be the following:
  - On initialization
    - $\mathcal{O}_0$ generates keys k_i, $i \in \mathbb{N}$;
    - $\mathcal{O}_1$ generates the key k.
  - On input (i, e) where e is an expression with free variables k_j the machine $\mathcal{O}_0$
    - evaluates e, letting k_j refer to its keys,
    - encrypts the result with k_i and returns it;

    and the machine $\mathcal{O}_1$ returns $\mathcal{E}(k, 0^{\text{const}})$.
- If no reasonably powerful adversary $\mathcal{A}$ can guess b from the interaction with $\mathcal{O}_b$ 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

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

- 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 τ are

$$\tau ::= int \mid key \mid enc(\tau) \mid (\tau, \tau)$$

- We may want to compute with ciphertexts, hence we subtype $enc(\tau) \leq int$.
- Types of operations:
  - arithmetic operations: $int^k \to int$;
  - pairing: $\tau_1 \times \tau_2 \to (\tau_1, \tau_2)$; i-th projection: $(\tau_1, \tau_2) \to \tau_i$;
  - key generation: $\mathbf{1} \to key$;
  - encryption: $key \times \tau \to enc(\tau)$;
    decryption: $key \times enc(\tau) \to \tau$;
  - guards: int.
- [AHS06] already has such a type system.

Removing decryptions

- Change the real-world program:
  - Give names to keys: replace each $k := \mathsf{newkey}$ with

$$k := \mathsf{newkey}; k_{\mathrm{name}} := c; c := c + 1$$

  - for each ciphertext record the key name and the plaintext in the auxiliary variables. Replace $y := \mathcal{E}(k, x)$ with

$$y := \mathcal{E}(k, x); y_{\mathrm{keyname}} := k_{\mathrm{name}}; y_{\mathrm{ptext}} := x$$

  - Replace the statements $x := \mathcal{D}(k, y)$ with

$$\mathbf{if}\ k_{\mathrm{name}} = y_{\mathrm{keyname}}\ \mathbf{then}\ x := y_{\mathrm{ptext}}\ \mathbf{else}\ x := \bot\ \mathbf{fi}$$

- The low-visible semantics does not change.

Encryption → random number gen.-tion

- Apply the definition of IND-KDM-CPA to the real-world program:
  - Replace each $\mathcal{E}(k, y)$ with $\mathcal{E}(k_0, 0)$.
- $\mathcal{E}(k_0, 0)$ 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

$$\mathbf{if}\ \mathsf{rnd}(\{0,1\}) = 1\ \mathbf{then}\ l := h\ \mathbf{else}\ l := \mathsf{rnd}(\{1,\ldots,100\})$$

- The possible values of l do not depend on h.
- But their distribution depends on h.
- We can come up with similiar examples in our language.
  - Using $\mathcal{E}$ in place of rnd.
- Hence using ciphertexts in computations is questionable as well.
- Remove the subtyping $enc(\tau) \leq int$.

The conditions for the program

- The variables are typed, as specified before.

$$\tau ::= int \mid key \mid enc(\tau) \mid (\tau, \tau)$$

  (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.