

Codes and Cryptography

Jorge L. Villar

MAMME, Fall 2015

PART XI

Defining a Security Notion

Defining security for a particular system requires:

- Defining the functionality of the system
- Defining the capabilities of the adversary
- Defining the goal of the adversary

The latter two can be captured by

- a random experiment (game) between a Challenger and the Adversary
- a special outcome indicating success of the Adversary
- a statement about the probability of that outcome

Outline

1 Defining Security

2 Proving Security

Example 1: One-Way Security

Assume that $\Pi = (\text{KeyGen}, \text{Enc}, \text{Dec})$ is a symmetric encryption scheme for the spaces $\mathcal{M}, \mathcal{C}, \mathcal{K}$ and security parameter ℓ .

Experiment $\text{Exp-SE-OW}(\Pi, \mathcal{A}, \ell)$:

$k \leftarrow \text{KeyGen}(\ell)$;

$m_* \leftarrow \mathcal{M}_\ell$;

$c_* \leftarrow \text{Enc}(k, m_*)$;

$m' \leftarrow \mathcal{A}(1^\ell, c_*)$;

if $m' = m_*$ **output** 1; // \mathcal{A} wins

else output 0;

The security statement is

Definition (SE-OW)

The symmetric encryption scheme Π is SE-OW secure if for all Probabilistic Polynomial-Time Turing Machine (PPTM), \mathcal{A} ,

$$\Pr[\text{Exp-SE-OW}(\Pi, \mathcal{A}, \ell) = 1] \in \text{negl}(\ell)$$

Example 2: Stronger Attacks

In some practical scenarios, an adversary has access to some pairs plaintext/ciphertext for the target key.

Experiment $\text{Exp-SE-OW-CPA}(\Pi, \mathcal{A}, \ell)$:

```

k ← KeyGen(ℓ);
m* ← Mℓ;
c* ← Enc(k, m*);
m' ← AOEnc(1ℓ, c*);
if m' = m* output 1; //A wins
else output 0;
    
```

Oracle $\mathcal{O}_{\text{Enc}}(m)$:
output $\text{Enc}(k, m)$;

Example 2: Stronger Attacks

In some practical scenarios, an adversary has access to some pairs plaintext/ciphertext for the target key.

Experiment $\text{Exp-SE-OW-CCA}(\Pi, \mathcal{A}, \ell)$:

```

k ← KeyGen(ℓ);
m* ← Mℓ;
c* ← Enc(k, m*);
m' ← AOEnc, ODec(1ℓ, c*);
if m' = m* output 1; //A wins
else output 0;
    
```

Oracle $\mathcal{O}_{\text{Enc}}(m)$:
output $\text{Enc}(k, m)$;

Oracle $\mathcal{O}_{\text{Dec}}(c)$:
if $c = c_*$ **output** \perp ; //Illegal oracle query
else output $\text{Dec}(k, c)$;

The number of queries q_{Enc} and q_{Dec} can be considered as additional security parameters

Example 2: Stronger Attacks

In some practical scenarios, an adversary has access to some pairs plaintext/ciphertext for the target key.

Experiment $\text{Exp-SE-OW-CCA}(\Pi, \mathcal{A}, \ell)$:

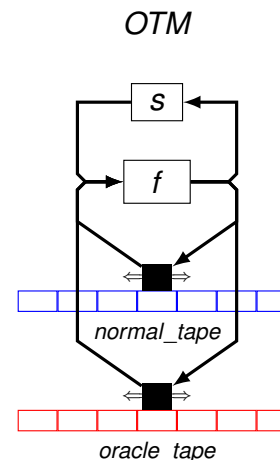
```

k ← KeyGen(ℓ);
m* ← Mℓ;
c* ← Enc(k, m*);
m' ← AOEnc, ODec(1ℓ, c*);
if m' = m* output 1; //A wins
else output 0;
    
```

Oracle $\mathcal{O}_{\text{Enc}}(m)$:
output $\text{Enc}(k, m)$;

Oracle $\mathcal{O}_{\text{Dec}}(c)$:
if $c = c_*$ **output** \perp ; //Illegal oracle query
else output $\text{Dec}(k, c)$;

Oracle Turing Machine



Special state: 'oracle_query'

The OTM enters in a waiting state until some external entity (not necessarily a Turing Machine) replaces the information in the oracle tape, in unit time.

NOTATION: $\text{OTM}^{\mathcal{O}}$

The oracle tape is used as a communication tape. Interactive Turing Machines can be defined following the same idea.

Example 3: Even Stronger Attacks

The adversary could have some a priori information about the target plaintext.

Experiment $\text{Exp-SE-LR}(\Pi, \mathcal{A}, \ell)$:

$k \leftarrow \text{KeyGen}(\ell)$;

$b_* \leftarrow \{0, 1\}$;

$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{LR}}}(1^\ell)$;

if $b' = b_*$ **output** 1; //A wins

else output 0;

Oracle $\mathcal{O}_{\text{LR}}(m_0, m_1)$:

if $\text{length}(m_0) \neq \text{length}(m_1)$ **output** \perp ; //Illegal oracle query

output $\text{Enc}(k, m_{b_*})$;

Outline

1 Defining Security

2 Proving Security

Example 3: Even Stronger Attacks

Definition (SE-LR)

The symmetric encryption scheme Π is SE-LR secure if for all Probabilistic Polynomial-Time Oracle Turing Machine (PPOTM), \mathcal{A} ,

$$|\Pr[\text{Exp-SE-LR}(\Pi, \mathcal{A}, \ell) = 1] - 1/2| \in \text{negl}(\ell)$$

The number of queries q_{LR} can be considered as an additional security parameter

The other notions SE-LR-CPA and SE-LR-CCA are defined accordingly

Translating Languages

Reduction: An efficient transformation $T : \{0, 1\}^* \rightarrow \{0, 1\}^*$ that maps a language L into another language L' , and also maps $\{0, 1\}^* \setminus L$ into $\{0, 1\}^* \setminus L'$.

NOTATION: $L \Rightarrow_{PP} L'$ or “ L reduces to L' ”

Definition (PP-Reduction of Languages)

A language L is PP-reducible to another language L' if there exists a PPTM T and a integer-valued function $q \in \text{poly}$ such that $T(\{0, 1\}^\ell) \subseteq \{0, 1\}^{q(\ell)}$, $T(L) \subseteq L'$ and $T(\{0, 1\}^* \setminus L) \subseteq \{0, 1\}^* \setminus L'$

Theorem

$L \notin \mathcal{BPP}$ and $L \Rightarrow_{PP} L'$ implies $L' \notin \mathcal{BPP}$

Reducing Computational Problems

Let P, P' be two (search/decision) problem families.

What's the meaning of " **P is hard on average implies P' is hard on average**"? Or equivalently, " **P' is not hard on average implies neither is P** "

" **P is not hard on average**" means there **exists** a PPTM with a non-negligible success probability/advantage in solving a random instance of P

Showing only the existence is a **non-constructive** proof. **Not meaningful in practice.**

Constructive proof: Explicitly (and efficiently) build a PPTM solving P from another PPTM solving P'

Black-Box Reductions

\mathcal{R} is just a Oracle PPTM and now $\mathcal{A} = \mathcal{R}[\mathcal{A}'] = \mathcal{R}^{\mathcal{A}'}$

- \mathcal{R} has no access to the internals of \mathcal{A}' , but only to its input-output behavior (**functionality**)
- Recall that \mathcal{A}' is non-perfect, i.e., it solves P' with a (**very small**) non-negligible probability/advantage
- \mathcal{R} can run several instances of \mathcal{A}' on different inputs, **but then it is hard to relate $\text{Succ}_{P',\mathcal{A}'}(\ell)$ and $\text{Succ}_{P,\mathcal{R}[\mathcal{A}']}(\ell)$**

A typical reduction: Black-Box with a single call to \mathcal{A}' :

- $\mathcal{R}[\mathcal{A}']$ transforms its input $x \in P$ into $x' \in P'$
- $\mathcal{R}[\mathcal{A}']$ runs \mathcal{A}' with input x'
- $\mathcal{R}[\mathcal{A}']$ computes its output from the output of \mathcal{A}'

Reducing Computational Problems

Constructive proofs for the statement $P \Rightarrow_{PP} P'$:

Give a PPTM \mathcal{R} that transforms (the description of) any PPTM \mathcal{A}' solving a random instance of P' into (the description of) another PPTM $\mathcal{A} = \mathcal{R}[\mathcal{A}']$ solving P such that

$$\text{Succ}_{P',\mathcal{A}'}(\ell) > \text{negl}(\ell) \Rightarrow \text{Succ}_{P,\mathcal{R}[\mathcal{A}']}(\ell) > \text{negl}(\ell)$$

where $\text{Succ}_{P,\mathcal{A}}(\ell)$ is $\Pr[\mathcal{A}(x) \in \text{sol}(x) : x \leftarrow P_\ell]$ for search problems, and

$$\left| \Pr[\mathcal{A}(x) = 1 : x \leftarrow L_P \cap \{0, 1\}^\ell] - \Pr[\mathcal{A}(x) = 1 : x \leftarrow \{0, 1\}^\ell \setminus L_P] \right|$$

for decision problems

Self-Reductions: An Example

Probability Amplification by Repetition is an example of Black-Box Self-Reduction of a decision problem

$\mathcal{R}[\mathcal{A}']$ runs n times \mathcal{A}' on the same input and decides its output by majority voting among the n outputs

For small $\text{Succ}_{P,\mathcal{A}'}(\ell)$

$$\text{Succ}_{P,\mathcal{R}[\mathcal{A}']}(\ell) \approx \sqrt{\frac{2n}{\pi}} \text{Succ}_{P,\mathcal{A}'}(\ell)$$

while $\text{time}(\mathcal{R}[\mathcal{A}'], x) \approx n \cdot \text{time}(\mathcal{A}', x)$

For ('checkable') search problems and small $\text{Succ}_{P,\mathcal{A}'}(\ell)$

$$\text{Succ}_{P,\mathcal{R}[\mathcal{A}']}(\ell) \approx n \text{Succ}_{P,\mathcal{A}'}(\ell)$$

and the meaningful quantity for comparisons is probability/time

Random Self-Reducibility

Definition

The decision problem family P is **random self-reducible** if there exists a PPTM T that transforms any particular instance $x \in P_\ell$ into a random (**uniform**) instance in P_ℓ .

T transforms any probability distribution in P_ℓ into the uniform

Using T as a self-reduction \mathcal{R}_T ,

$$\mathcal{A}(x) = \mathcal{R}_T[\mathcal{A}'](x) = \mathcal{A}'(T(x))$$

proves that solving a random instance of P is not easier than (thus, **equivalent to**) solving **all** instances in P .

For a random self-reducible problem **average hardness is equivalent to worst-case hardness**

A Sample Reduction: SE-LR-CPA \Rightarrow SE-OW-CPA

Experiment

Exp-SE-LR-CPA(Π, \mathcal{A}, ℓ):

$k \leftarrow \text{KeyGen}(\ell);$
 $b_* \leftarrow \{0, 1\};$
 $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{LR}}, \mathcal{O}_{\text{Enc}}}(1^\ell);$
if $b' = b_*$ **output** 1;
else output 0;

Oracle $\mathcal{O}_{\text{LR}}(m_0, m_1)$:
if $|m_0| \neq |m_1|$
output \perp ;
else
output $\text{Enc}(k, m_{b_*});$

Oracle $\mathcal{O}_{\text{Enc}}(m)$:
output $\text{Enc}(k, m);$

Reduction:

$m_0, m_1 \leftarrow \mathcal{M}_\ell;$
 $c_* \leftarrow \mathcal{O}_{\text{LR}}(m_0, m_1);$
 $m' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Enc}}}(1^\ell, c_*);$
if $m' = m_1$
output 1;
else if $m' = m_0$
output 0;
else
output $b' \leftarrow \{0, 1\};$

Experiment

Exp-SE-OW-CPA(Π, \mathcal{A}, ℓ):

$k \leftarrow \text{KeyGen}(\ell);$
 $m_* \leftarrow \mathcal{M}_\ell;$
 $c_* \leftarrow \text{Enc}(k, m_*);$
 $m' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Enc}}}(1^\ell, c_*);$
if $m' = m_*$
output 1;
else output 0;

Oracle $\mathcal{O}_{\text{Enc}}(m)$:
output $\text{Enc}(k, m);$

Applications of Reductions (I)

Recall that security definitions are stated as (interactive) problem families.

Reductions between security notions show implications, or relative hardness, e.g.,

→ details...

$$\text{SE-LR-CCA} \Rightarrow \text{SE-LR-CPA} \Rightarrow \text{SE-OW-CPA} \Rightarrow \text{SE-OW}$$

(strongest)

(weakest)

A reduction \mathcal{R} from a security notion SEC1 into another notion SEC2 transforms an adversary \mathcal{A}_2 breaking SEC2 into another $\mathcal{A}_1 = \mathcal{R}[\mathcal{A}_2]$ breaking SEC1.

Thus, \mathcal{R} **simulates** any oracle given in SEC2 for \mathcal{A}_2 , but it can **use** the oracles given in SEC1.

Applications of Reductions (II)

Reductions between computational problems show relative strongness of the different security assumptions, e.g., for a cyclic group G ,

$$\text{DDH}\langle G \rangle \Rightarrow \text{CDH}\langle G \rangle \Rightarrow \text{DLOG}\langle G \rangle$$

(strongest)

(weakest)

Security proofs by reduction: A reduction of a computational problem family P to the problem of breaking a security notion SEC for a cryptosystem Π , proves security of Π under the assumption that P is hard

$$P \Rightarrow \text{SEC}\langle \Pi \rangle$$

It reads “if someone breaks Π , he also solves P ”

Provable Security

Main goal in provable security: Give a proof by reduction under a well-known and well-studied assumption

- The same assumption can be used for several cryptosystems
- ... even if they are of different type (e.g., encryption and signatures)
- It makes easier comparing them
- Cryptanalysis focus on computational problems and not on specific schemes

...but some reductions are not meaningful in practice...

A Remark About Tightness

$P \Rightarrow \text{SEC}(\Pi)$ reads “if someone breaks Π , he also solves P ”

More precisely, “there exists \mathcal{R} such that if \mathcal{A} breaks Π in time t_1 with probability/advantage $\varepsilon_1 > \text{negl}(\ell)$, then $\mathcal{R}[\mathcal{A}]$ solves P in time t_2 with probability/advantage $\varepsilon_2 > \text{negl}(\ell)$ ”

- If $t_2 \approx t_1$ and $\varepsilon_2 \approx \varepsilon_1$, \mathcal{R} is **tight Meaningful reduction!**
- If $t_2 \approx t_1$ but $\varepsilon_2 \approx C\varepsilon_1$ for some constant $C \ll 1$, \mathcal{R} is **almost tight Quite meaningful reduction!**
- If $t_2 \approx t_1$ but $\varepsilon_2/\varepsilon_1 \rightarrow 0$ as $\ell \rightarrow \infty$, \mathcal{R} is almost **not tight It depends...**
- If $t_2 \gg t_1$, compare the ratios ε_1/t_1 and ε_2/t_2

Codes and Cryptography

Jorge L. Villar

MAMME, Fall 2015

END OF PART XI