Adaptive Security of Symbolic Encryption: the Case of Dynamic Corruptions.

Laurent Mazaré
Joint work with Bogdan Warinschi

LSV - ENS Cachan

Formacrypt Metting, January 2007

Introduction

Symbolic and Computational Settings

1. Symbolic model: automatic verification.
2. Computational model: realism.

Computational soundness of symbolic encryption

Symbolic security $\Rightarrow$ Computational security.

Adversary Model

1. Passive adversary: eavesdropper.
2. Adaptive adversaries.
3. Active adversaries.

Introduction

Symbolic and Computational Settings

1. Symbolic model: automatic verification.
2. Computational model: realism.

Computational soundness of symbolic encryption

Symbolic security $\Rightarrow$ Computational security.

Adversary Model

1. Passive adversary: eavesdropper.
2. Adaptive adversaries.
3. Active adversaries.

Introduction

Symbolic and Computational Settings

1. Symbolic model: automatic verification.
2. Computational model: realism.

Computational soundness of symbolic encryption

Symbolic security $\Rightarrow$ Computational security.

Adversary Model

1. Passive adversary: eavesdropper.
2. Adaptive adversaries.
3. Active adversaries.

Structure

1. Passive Soundness
2. Adaptive Soundness
3. Selective Decryption

Messages and Deductions

Messages

Messages are defined by the following grammar:

$$msg ::= (msg, msg) \mid \{msg\}_k \mid k \mid 0 \mid 1$$

Dolev-Yao Deduction System

Message m is deducible from the set of messages E, $E \vdash m$ if:

$$\frac{m \in E}{E \vdash m}\,(\text{Ax}) \qquad \frac{E \vdash (m_1, m_2)}{E \vdash m_1}\,(\text{Pr1}) \qquad \frac{E \vdash (m_1, m_2)}{E \vdash m_2}\,(\text{Pr2})$$

$$\frac{E \vdash m_1 \quad E \vdash m_2}{E \vdash (m_1, m_2)}\,(\text{P}) \qquad \frac{E \vdash m \quad E \vdash k}{E \vdash \{m\}_k}\,(\text{Ch}) \qquad \frac{E \vdash \{m\}_k \quad E \vdash k}{E \vdash m}\,(\text{Déc})$$

Messages and Deductions

Messages

Messages are defined by the following grammar:

$$msg ::= (msg, msg) \mid \{msg\}_k \mid k \mid 0 \mid 1$$

Dolev-Yao Deduction System

Message m is deducible from the set of messages E, $E \vdash m$ if:

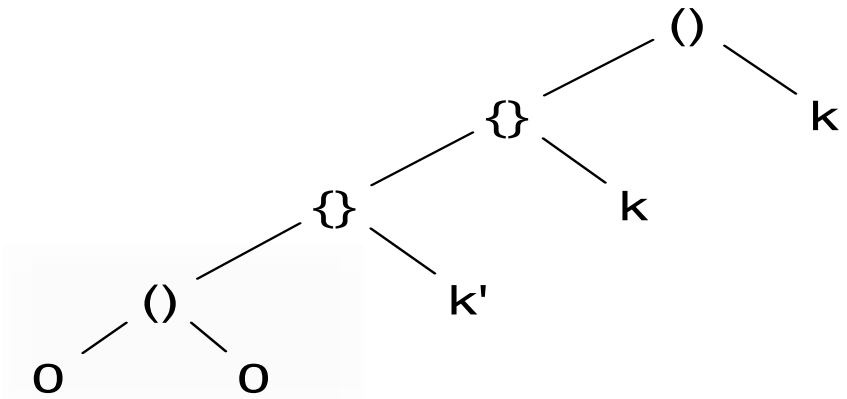
$$\frac{m \in E}{E \vdash m}(\text{Ax}) \qquad \frac{E \vdash (m_1, m_2)}{E \vdash m_1}(\text{Pr1}) \qquad \frac{E \vdash (m_1, m_2)}{E \vdash m_2}(\text{Pr2})$$

$$\frac{E \vdash m_1 \quad E \vdash m_2}{E \vdash (m_1, m_2)}(\text{P}) \qquad \frac{E \vdash m \quad E \vdash k}{E \vdash \{m\}_k}(\text{Ch}) \qquad \frac{E \vdash \{m\}_k \quad E \vdash k}{E \vdash m}(\text{Déc})$$

Accessible information from a message

Pattern

- $m = (\{\{0,0\}_{k'}\}_k, k)$
- Pattern : accessible information.



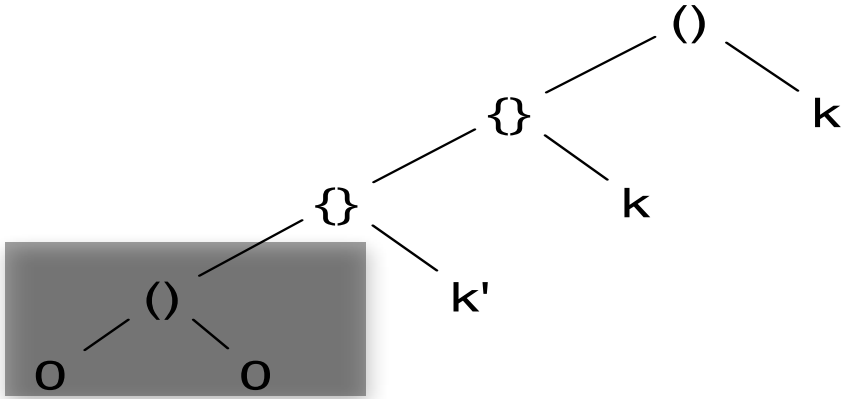
Equivalence

- $n = (\{\{1,1\}_{k'}\}_k, k)$
- m and n are indistinguishable.

Accessible information from a message

Pattern

- $m = (\{\{0,0\}_{k'}\}_k, k)$
- Pattern : accessible information.



Equivalence

- $n = (\{\{1,1\}_{k'}\}_k, k)$
- m and n are indistinguishable.

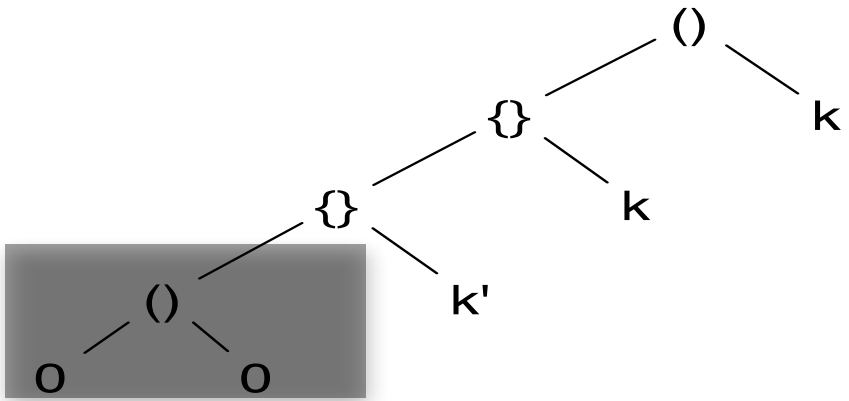
Accessible information from a message

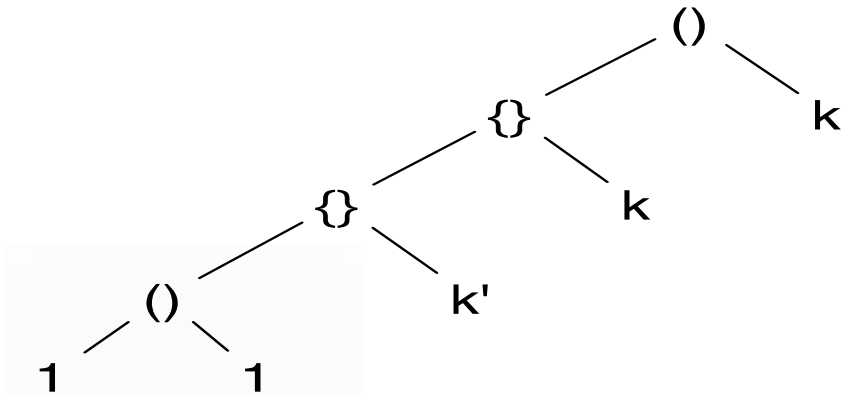
Pattern

- $m = (\{\{0,0\}_{k'}\}_k, k)$
- Pattern : accessible information.

Equivalence

- $n = (\{\{1,1\}_{k'}\}_k, k)$
- m and n are indistinguishable.





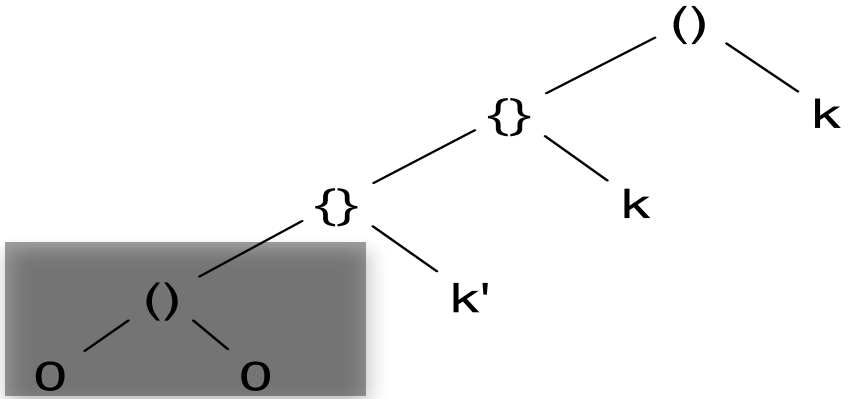
Accessible information from a message

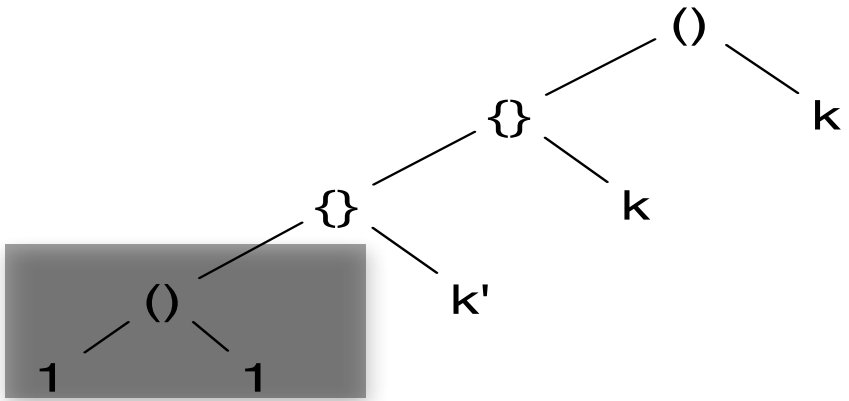
Pattern

- $m = (\{\{0,0\}_{k'}\}_k, k)$
- Pattern : accessible information.

Equivalence

- $n = (\{\{1,1\}_{k'}\}_k, k)$
- m and n are indistinguishable.





Symbolic Equivalence

Patterns

$$\begin{aligned}
\mathrm{pat}((m_1, m_2), K) &= (\mathrm{pat}(m_1, K), \mathrm{pat}(m_2, K)) \\
\mathrm{pat}(\{m'\}_k, K) &= \{\mathrm{pat}(m', K)\}_k && \text{if } k \in K \\
\mathrm{pat}(\{m'\}_k, K) &= \{\square\}_k && \text{if } k \notin K \\
\mathrm{pat}(k, K) &= k
\end{aligned}$$

Symbolic Equivalence

- m is equivalent to n ($m \equiv n$) if

$$\mathrm{pat}(m, \{k / m \vdash k\}) = \mathrm{pat}(n, \{k / n \vdash k\})$$

- m is equivalent to n up to renaming ($m \cong n$) if

$$\exists \sigma \text{ permutation of keys}, m \equiv n\sigma$$

Symbolic Equivalence

Patterns

$$\begin{aligned}
\mathrm{pat}((m_1, m_2), K) &= (\mathrm{pat}(m_1, K), \mathrm{pat}(m_2, K)) \\
\mathrm{pat}(\{m'\}_k, K) &= \{\mathrm{pat}(m', K)\}_k && \text{if } k \in K \\
\mathrm{pat}(\{m'\}_k, K) &= \{\square\}_k && \text{if } k \notin K \\
\mathrm{pat}(k, K) &= k
\end{aligned}$$

Symbolic Equivalence

- m is equivalent to n ($m \equiv n$) if

$$\mathrm{pat}(m, \{k/m \vdash k\}) = \mathrm{pat}(n, \{k/n \vdash k\})$$

- m is equivalent to n up to renaming ($m \cong n$) if

$$\exists \sigma \text{ permutation of keys}, m \equiv n\sigma$$

Symbolic Equivalence

Patterns

$$\begin{aligned}
\mathrm{pat}((m_1, m_2), K) &= (\mathrm{pat}(m_1, K), \mathrm{pat}(m_2, K)) \\
\mathrm{pat}(\{m'\}_k, K) &= \{\mathrm{pat}(m', K)\}_k && \text{if } k \in K \\
\mathrm{pat}(\{m'\}_k, K) &= \{\square\}_k && \text{if } k \notin K \\
\mathrm{pat}(k, K) &= k
\end{aligned}$$

Symbolic Equivalence

- m is equivalent to n ($m \equiv n$) if

$$\mathrm{pat}(m, \{k/m \vdash k\}) = \mathrm{pat}(n, \{k/n \vdash k\})$$

- m is equivalent to n up to renaming ($m \cong n$) if

$$\exists \sigma \text{ permutation of keys}, m \equiv n\sigma$$

Symbolic Equivalence

Patterns

$$\begin{aligned}
\mathrm{pat}((m_1, m_2), K) &= (\mathrm{pat}(m_1, K), \mathrm{pat}(m_2, K)) \\
\mathrm{pat}(\{m'\}_k, K) &= \{\mathrm{pat}(m', K)\}_k && \text{if } k \in K \\
\mathrm{pat}(\{m'\}_k, K) &= \{\Box\}_k && \text{if } k \notin K \\
\mathrm{pat}(k, K) &= k
\end{aligned}$$

Symbolic Equivalence

- m is equivalent to n ($m \equiv n$) if

$$\mathrm{pat}(m, \{k/m \vdash k\}) = \mathrm{pat}(n, \{k/n \vdash k\})$$

- m is equivalent to n up to renaming ($m \cong n$) if

$$\exists \sigma \text{ permutation of keys}, m \equiv n\sigma$$

Examples

Examples

1. Key renaming:
$$k \cong k'$$
2. Impossible renaming:
$$(k, k') \ncong (k, k)$$
3. Encryption without key:
$$\{0\}_k \cong \{1\}_k$$
4. Key revelation:
$$(\{1\}_{k'}, \{0\}_k) \ncong (\{1\}_k, \{0\}_k)$$
5. Non-deterministic encryption:
$$(\{0\}_k, \{0\}_k) \cong (\{0\}_k, \{1\}_k)$$

Examples

Examples

1. Key renaming:
$$k \cong k'$$
2. Impossible renaming:
$$(k, k') \ncong (k, k)$$
3. Encryption without key:
$$\{0\}_k \cong \{1\}_k$$
4. Key revelation:
$$(\{1\}_{k'}, \{0\}_k) \ncong (\{1\}_k, \{0\}_k)$$
5. Non-deterministic encryption:
$$(\{0\}_k, \{0\}_k) \cong (\{0\}_k, \{1\}_k)$$

Examples

Examples

1. Key renaming:
$$k \cong k'$$
2. Impossible renaming:
$$(k, k') \ncong (k, k)$$
3. Encryption without key:
$$\{0\}_k \cong \{1\}_k$$
4. Key revelation:
$$(\{1\}_{k'}, \{0\}_k) \ncong (\{1\}_k, \{0\}_k)$$
5. Non-deterministic encryption:
$$(\{0\}_k, \{0\}_k) \cong (\{0\}_k, \{1\}_k)$$

Examples

Examples

1. Key renaming:
$$k \cong k'$$
2. Impossible renaming:
$$(k, k') \not\cong (k, k)$$
3. Encryption without key:
$$\{0\}_k \cong \{1\}_k$$
4. Key revelation:
$$(\{1\}_{k'}, \{0\}_k) \not\cong (\{1\}_k, \{0\}_k)$$
5. Non-deterministic encryption:
$$(\{0\}_k, \{0\}_k) \cong (\{0\}_k, \{1\}_k)$$

Examples

Examples

1. Key renaming:
$$k \cong k'$$
2. Impossible renaming:
$$(k, k') \not\cong (k, k)$$
3. Encryption without key:
$$\{0\}_k \cong \{1\}_k$$
4. Key revelation:
$$(\{1\}_{k'}, \{0\}_k) \not\cong (\{1\}_k, \{0\}_k)$$
5. Non-deterministic encryption:
$$(\{0\}_k, \{0\}_k) \cong (\{0\}_k, \{1\}_k)$$

Semantic Security

Symmetric Encryption Scheme

A symmetric encryption scheme Π is made of three algorithms:

- $\mathcal{KG}$: key generation, $\eta \longrightarrow k$.
- $\mathcal{E}$: encryption, $bs, k \longrightarrow bs$.
- $\mathcal{D}$: decryption, $bs, k \longrightarrow bs$.

Security against Chosen Plaintext Attacks

Scheme $(\mathcal{KG}, \mathcal{E}, \mathcal{D})$ is IND-CPA secure if for any adversary $\mathcal{A}$,

$$Pr\left[\begin{array}{l} k \leftarrow \mathcal{KG} \\ \mathcal{A}^{(bs_0, bs_1) \rightarrow \mathcal{E}(\mathbf{bs_1}, k) = 1} \end{array}\right] - Pr\left[\begin{array}{l} k \leftarrow \mathcal{KG} \\ \mathcal{A}^{(bs_0, bs_1) \rightarrow \mathcal{E}(\mathbf{bs_0}, k) = 1} \end{array}\right]$$

is negligible in η (lower than $1/\eta^c$ for any c).

Semantic Security

Symmetric Encryption Scheme

A symmetric encryption scheme Π is made of three algorithms:

- $\mathcal{KG}$: key generation, $\eta \longrightarrow k$.
- $\mathcal{E}$: encryption, $bs, k \longrightarrow bs$.
- $\mathcal{D}$: decryption, $bs, k \longrightarrow bs$.

Security against Chosen Plaintext Attacks

Scheme $(\mathcal{KG}, \mathcal{E}, \mathcal{D})$ is IND-CPA secure if for any adversary $\mathcal{A}$,

$$Pr\left[\begin{array}{l} k \leftarrow \mathcal{KG} \\ \mathcal{A}^{(bs_0, bs_1) \rightarrow \mathcal{E}(\mathbf{bs_1}, k)=1} \end{array}\right] - Pr\left[\begin{array}{l} k \leftarrow \mathcal{KG} \\ \mathcal{A}^{(bs_0, bs_1) \rightarrow \mathcal{E}(\mathbf{bs_0}, k)=1} \end{array}\right]$$

is negligible in η (lower than $1/\eta^c$ for any c).

Computational Soundness

Indistingable Distributions

Two distributions D_0 et D_1 ($\eta \longrightarrow bs$) are indistinguishable, $D_0 \approx D_1$, if for any $\mathcal{A}$,

$$Pr\left[\begin{array}{l} bs \leftarrow \mathbf{D_1} \\ \mathcal{A}(bs) = 1 \end{array}\right] - Pr\left[\begin{array}{l} bs \leftarrow \mathbf{D_0} \\ \mathcal{A}(bs) = 1 \end{array}\right]$$

is negligible in η.

Theorem (Soundness in the Passive Case)

If m and n are key acyclic and Π is IND-CPA, then

$$m \cong n \Rightarrow [\![m]\!]_\Pi \approx [\![n]\!]_\Pi$$

Computational Soundness

Indistingable Distributions

Two distributions D_0 et D_1 ($\eta \longrightarrow bs$) are indistinguishable, $D_0 \approx D_1$, if for any $\mathcal{A}$,

$$Pr\left[\begin{array}{l} bs \leftarrow \mathbf{D_1} \\ \mathcal{A}(bs) = 1 \end{array}\right] - Pr\left[\begin{array}{l} bs \leftarrow \mathbf{D_0} \\ \mathcal{A}(bs) = 1 \end{array}\right]$$

is negligible in η.

Theorem (Soundness in the Passive Case)

If m and n are key acyclic and Π is IND-CPA, then

$$m \cong n \Rightarrow [\![m]\!]_\Pi \approx [\![n]\!]_\Pi$$

Structure

1 Passive Soundness

2 Adaptive Soundness

3 Selective Decryption

The Adaptive Model

Model

- Intuition: $\mathcal{A}$ chooses (m_0^1, m_1^1) and obtains $[\![m_b^1]\!]_\Pi$
- Then $\mathcal{A}$ chooses (m_0^2, m_1^2) and obtains $[\![m_b^2]\!]_\Pi$...
- An adversary has to guess a bit b using an oracle $\mathcal{O}$.

$$\mathcal{O}_{b,\Pi,\eta} \; : \; (m_0, m_1) \rightarrow [\![m_b]\!]_\Pi$$

Definition

The advantage of $\mathcal{A}$ against Adaptive Security of Π is:

$$Adv^{ad}_{\mathcal{A},\Pi}(\eta) = Pr[\mathcal{A}^{\mathcal{O}_{1,\Pi,\eta}} = 1] - Pr[\mathcal{A}^{\mathcal{O}_{0,\Pi,\eta}} = 1]$$

The Adaptive Model

Model

- Intuition: $\mathcal{A}$ chooses (m_0^1, m_1^1) and obtains $[\![m_b^1]\!]_\Pi$
- Then $\mathcal{A}$ chooses (m_0^2, m_1^2) and obtains $[\![m_b^2]\!]_\Pi$...
- An adversary has to guess a bit b using an oracle $\mathcal{O}$.

$$\mathcal{O}_{b,\Pi,\eta} : (m_0, m_1) \rightarrow [\![m_b]\!]_\Pi$$

Definition

The advantage of $\mathcal{A}$ against Adaptive Security of Π is:

$$Adv^{ad}_{\mathcal{A},\Pi}(\eta) = Pr[\mathcal{A}^{\mathcal{O}_{1,\Pi,\eta}} = 1] - Pr[\mathcal{A}^{\mathcal{O}_{0,\Pi,\eta}} = 1]$$

The Adaptive Model

Model

- Intuition: $\mathcal{A}$ chooses (m_0^1, m_1^1) and obtains $[\![m_b^1]\!]_\Pi$
- Then $\mathcal{A}$ chooses (m_0^2, m_1^2) and obtains $[\![m_b^2]\!]_\Pi$...
- An adversary has to guess a bit b using an oracle $\mathcal{O}$.

$$\mathcal{O}_{b,\Pi,\eta} \; : \; (m_0, m_1) \rightarrow [\![m_b]\!]_\Pi$$

Definition

The advantage of $\mathcal{A}$ against Adaptive Security of Π is:

$$Adv^{ad}_{\mathcal{A},\Pi}(\eta) = Pr[\mathcal{A}^{\mathcal{O}_{1,\Pi,\eta}} = 1] - Pr[\mathcal{A}^{\mathcal{O}_{0,\Pi,\eta}} = 1]$$

The Adaptive Model

Model

- Intuition: $\mathcal{A}$ chooses (m_0^1, m_1^1) and obtains $[\![m_b^1]\!]_\Pi$
- Then $\mathcal{A}$ chooses (m_0^2, m_1^2) and obtains $[\![m_b^2]\!]_\Pi$...
- An adversary has to guess a bit b using an oracle $\mathcal{O}$.

$$\mathcal{O}_{b,\Pi,\eta} \ : \ (m_0, m_1) \rightarrow [\![m_b]\!]_\Pi$$

Definition

The advantage of $\mathcal{A}$ against Adaptive Security of Π is:

$$Adv^{ad}_{\mathcal{A},\Pi}(\eta) = Pr[\mathcal{A}^{\mathcal{O}_{1,\Pi,\eta}} = 1] - Pr[\mathcal{A}^{\mathcal{O}_{0,\Pi,\eta}} = 1]$$

Examples

Examples

Easy ways to win:

1. $\mathcal{A}$ makes query $\mathcal{O}(0,1)$, receives b, $Adv^{ad}_{\mathcal{A},\Pi}(\eta) = 1$
   Restriction: queries have to be equivalent.
2. $\mathcal{A}$ makes query $\mathcal{O}(\{0\}_k, \{1\}_k)$, receives $\{b\}_k$,
   then makes query $\mathcal{O}(k,k)$ and obtains b, $Adv^{ad}_{\mathcal{A},\Pi}(\eta) = 1$
   Restriction: sequence of queries have to be equivalent.
3. $\mathcal{A}$ makes query $\mathcal{O}(\{k\}_k, \{0\}_k)$ and receives b,
   Restriction: key-acyclicity of queries.

New possibilities of the adversary:

1. $\mathcal{A}$ makes query $\mathcal{O}(\{0\}_k, \{0\}_k)$, receives $\{0\}_k$,
   $\mathcal{A}$ uses this information to obtain b.

Examples

Examples

Easy ways to win:

1. $\mathcal{A}$ makes query $\mathcal{O}(0,1)$, receives b, $Adv^{ad}_{\mathcal{A},\Pi}(\eta) = 1$
   Restriction: queries have to be equivalent.
2. $\mathcal{A}$ makes query $\mathcal{O}(\{0\}_k, \{1\}_k)$, receives $\{b\}_k$,
   then makes query $\mathcal{O}(k,k)$ and obtains b, $Adv^{ad}_{\mathcal{A},\Pi}(\eta) = 1$
   Restriction: sequence of queries have to be equivalent.
3. $\mathcal{A}$ makes query $\mathcal{O}(\{k\}_k, \{0\}_k)$ and receives b,
   Restriction: key-acyclicity of queries.

New possibilities of the adversary:

1. $\mathcal{A}$ makes query $\mathcal{O}(\{0\}_k, \{0\}_k)$, receives $\{0\}_k$,
   $\mathcal{A}$ uses this information to obtain b.

Examples

Examples

Easy ways to win:

1. $\mathcal{A}$ makes query $\mathcal{O}(0,1)$, receives b, $Adv^{ad}_{\mathcal{A},\Pi}(\eta) = 1$
   Restriction: queries have to be equivalent.
2. $\mathcal{A}$ makes query $\mathcal{O}(\{0\}_k, \{1\}_k)$, receives $\{b\}_k$,
   then makes query $\mathcal{O}(k,k)$ and obtains b, $Adv^{ad}_{\mathcal{A},\Pi}(\eta) = 1$
   Restriction: sequence of queries have to be equivalent.
3. $\mathcal{A}$ makes query $\mathcal{O}(\{k\}_k, \{0\}_k)$ and receives b,
   Restriction: key-acyclicity of queries.

New possibilities of the adversary:

1. $\mathcal{A}$ makes query $\mathcal{O}(\{0\}_k, \{0\}_k)$, receives $\{0\}_k$,
   $\mathcal{A}$ uses this information to obtain b.

Examples

Examples

Easy ways to win:

1. $\mathcal{A}$ makes query $\mathcal{O}(0,1)$, receives b, $Adv^{ad}_{\mathcal{A},\Pi}(\eta) = 1$
   Restriction: queries have to be equivalent.
2. $\mathcal{A}$ makes query $\mathcal{O}(\{0\}_k, \{1\}_k)$, receives $\{b\}_k$,
   then makes query $\mathcal{O}(k,k)$ and obtains b, $Adv^{ad}_{\mathcal{A},\Pi}(\eta) = 1$
   Restriction: sequence of queries have to be equivalent.
3. $\mathcal{A}$ makes query $\mathcal{O}(\{k\}_k, \{0\}_k)$ and receives b,
   Restriction: key-acyclicity of queries.

New possibilities of the adversary:

1. $\mathcal{A}$ makes query $\mathcal{O}(\{0\}_k, \{0\}_k)$, receives $\{0\}_k$,
   $\mathcal{A}$ uses this information to obtain b.

Computational Soundness (Micciancio-Panjwani'04)

Restrictions over adversaries

If $\mathcal{A}$ performs queries $(m_0^1, m_1^1), \ldots, (m_0^\ell, m_1^\ell)$, then

1. Requests must not contain any key cycles:
   $(m_0^1, ..., m_0^\ell)$ and $(m_1^1, ..., m_1^\ell)$ are acyclic
2. Requests must be symbolically equivalent:
   $(m_0^1, ..., m_0^\ell) \cong (m_1^1, ..., m_1^\ell)$
3. Keys are sent then used:
   $k \in \text{plaintext}(m_b^i) \Rightarrow k \notin \text{key}(m_b^j)$ for $j < i$

Theorem

If Π is IND-CPA secure then for any legal adversary $\mathcal{A}$,

$Adv^{ad}_{\mathcal{A},\Pi}$ is negligible in η

Computational Soundness (Micciancio-Panjwani'04)

Restrictions over adversaries

If $\mathcal{A}$ performs queries $(m_0^1, m_1^1), \ldots, (m_0^\ell, m_1^\ell)$, then

1. Requests must not contain any key cycles:
   $(m_0^1, ..., m_0^\ell)$ and $(m_1^1, ..., m_1^\ell)$ are acyclic
2. Requests must be symbolically equivalent:
   $(m_0^1, ..., m_0^\ell) \cong (m_1^1, ..., m_1^\ell)$
3. Keys are sent then used:
   $k \in \text{plaintext}(m_b^i) \Rightarrow k \notin \text{key}(m_b^j)$ for $j < i$

Theorem

If Π is IND-CPA secure then for any legal adversary $\mathcal{A}$,

$Adv^{ad}_{\mathcal{A},\Pi}$ is negligible in η

Computational Soundness (Micciancio-Panjwani'04)

Restrictions over adversaries

If $\mathcal{A}$ performs queries $(m_0^1, m_1^1), \ldots, (m_0^\ell, m_1^\ell)$, then

1. Requests must not contain any key cycles:
   $(m_0^1, ..., m_0^\ell)$ and $(m_1^1, ..., m_1^\ell)$ are acyclic
2. Requests must be symbolically equivalent:
   $(m_0^1, ..., m_0^\ell) \cong (m_1^1, ..., m_1^\ell)$
3. Keys are sent then used:
   $k \in \text{plaintext}(m_b^i) \Rightarrow k \notin \text{key}(m_b^j)$ for $j < i$

Theorem

If Π is IND-CPA secure then for any legal adversary $\mathcal{A}$,

$Adv^{ad}_{\mathcal{A},\Pi}$ is negligible in η

Computational Soundness (Micciancio-Panjwani'04)

Restrictions over adversaries

If $\mathcal{A}$ performs queries $(m_0^1, m_1^1), \ldots, (m_0^\ell, m_1^\ell)$, then

1. Requests must not contain any key cycles:
   $(m_0^1, ..., m_0^\ell)$ and $(m_1^1, ..., m_1^\ell)$ are acyclic
2. Requests must be symbolically equivalent:
   $(m_0^1, ..., m_0^\ell) \cong (m_1^1, ..., m_1^\ell)$
3. Keys are sent then used:
   $k \in \text{plaintext}(m_b^i) \Rightarrow k \notin \text{key}(m_b^j)$ for $j < i$

Theorem

If Π is IND-CPA secure then for any legal adversary $\mathcal{A}$,

$Adv^{ad}_{\mathcal{A},\Pi}$ is negligible in η

Computational Soundness (Micciancio-Panjwani'04)

Restrictions over adversaries

If $\mathcal{A}$ performs queries $(m_0^1, m_1^1), \ldots, (m_0^\ell, m_1^\ell)$, then

1. Requests must not contain any key cycles:
$$(m_0^1, \ldots, m_0^\ell) \text{ and } (m_1^1, \ldots, m_1^\ell) \text{ are acyclic}$$
2. Requests must be symbolically equivalent:
$$(m_0^1, \ldots, m_0^\ell) \cong (m_1^1, \ldots, m_1^\ell)$$
3. Keys are sent then used:
$$k \in \text{plaintext}(m_b^i) \Rightarrow k \notin \text{key}(m_b^j) \text{ for } j < i$$

Theorem

If Π is IND-CPA secure then for any legal adversary $\mathcal{A}$,

$$Adv^{ad}_{\mathcal{A},\Pi} \text{ is negligible in } \eta$$

Static Corruption Model

The send-then-use hypothesis

- Keys are sent then use.

$$k \in \text{plaintext}(m_b^i) \Rightarrow k \notin \text{key}(m_b^j) \text{ for } j < i$$

- Use key k: $\mathcal{O}(\{0\}_k, \{0\}_k)$.
- Corrupt key k: $\mathcal{O}(k, k)$.
- send-then-use $\Rightarrow$ Static corruption model.

Is it possible to weaken/suppress this hypothesis and still have computational soundness ?

Dynamic corruption model

Static Corruption Model

The send-then-use hypothesis

- Keys are sent then use.

$$k \in \text{plaintext}(m_b^i) \Rightarrow k \notin \text{key}(m_b^j) \text{ for } j < i$$

- Use key k: $\mathcal{O}(\{0\}_k, \{0\}_k)$.
- Corrupt key k: $\mathcal{O}(k, k)$.
- send-then-use $\Rightarrow$ Static corruption model.

Is it possible to weaken/suppress this hypothesis and still have computational soundness ?

Dynamic corruption model

Static Corruption Model

The send-then-use hypothesis

- Keys are sent then use.

$$k \in \text{plaintext}(m_b^i) \Rightarrow k \notin \text{key}(m_b^j) \text{ for } j < i$$

- Use key k: $\mathcal{O}(\{0\}_k, \{0\}_k)$.
- Corrupt key k: $\mathcal{O}(k, k)$.
- send-then-use $\Rightarrow$ Static corruption model.

Is it possible to weaken/suppress this hypothesis and still have computational soundness ?

Dynamic corruption model

Static Corruption Model

The send-then-use hypothesis

- Keys are sent then use.

$$k \in \text{plaintext}(m_b^i) \Rightarrow k \notin \text{key}(m_b^j) \text{ for } j < i$$

- Use key k: $\mathcal{O}(\{0\}_k, \{0\}_k)$.
- Corrupt key k: $\mathcal{O}(k, k)$.
- send-then-use $\Rightarrow$ Static corruption model.

Is it possible to weaken/suppress this hypothesis and still have computational soundness ?

Dynamic corruption model

Static Corruption Model

The send-then-use hypothesis

- Keys are sent then use.

$$k \in \text{plaintext}(m_b^i) \Rightarrow k \notin \text{key}(m_b^j) \text{ for } j < i$$

- Use key k: $\mathcal{O}(\{0\}_k, \{0\}_k)$.
- Corrupt key k: $\mathcal{O}(k, k)$.
- send-then-use $\Rightarrow$ Static corruption model.

Is it possible to weaken/suppress this hypothesis and still have computational soundness ?

Dynamic corruption model

Structure

1 Passive Soundness

2 Adaptive Soundness

3 Selective Decryption

The Selective Decryption Problem

Formulation of M. Naor, 1995

- Let Π be an IND-CPA secure encryption scheme.
- Generate $p(\eta)$ independent keys k_i using $\mathcal{KG}$.
- Distribute a CDROM with $p(\eta)$ images N_i encrypted with k_i.
- Clients can purchase k_i via a website.
- If a client purchase several keys k_i for $i \in I$,

  can he deduce anything on the N_j for $j \notin I$?

Intuitive answer: no because keys are independent.

Instance of a more complex problem: Selective Decommitment linked to other open problems in cryptography.

The Selective Decryption Problem

Formulation of M. Naor, 1995

- Let Π be an IND-CPA secure encryption scheme.
- Generate $p(\eta)$ independent keys k_i using $\mathcal{KG}$.
- Distribute a CDROM with $p(\eta)$ images N_i encrypted with k_i.
- Clients can purchase k_i via a website.
- If a client purchase several keys k_i for $i \in I$,

  can he deduce anything on the N_j for $j \notin I$?

Intuitive answer: no because keys are independent.

Instance of a more complex problem: Selective Decommitment linked to other open problems in cryptography.

The Selective Decryption Problem

Formulation of M. Naor, 1995

- Let Π be an IND-CPA secure encryption scheme.
- Generate $p(\eta)$ independent keys k_i using $\mathcal{KG}$.
- Distribute a CDROM with $p(\eta)$ images N_i encrypted with k_i.
- Clients can purchase k_i via a website.
- If a client purchase several keys k_i for $i \in I$,

  can he deduce anything on the N_j for $j \notin I$?

Intuitive answer: no because keys are independent.

Instance of a more complex problem: Selective Decommitment linked to other open problems in cryptography.

In The Adaptive Model

Examples

We can map problems similar to selective decryption:

1. $\mathcal{A}$ makes query $\mathcal{O}\left(\{0\}_{k_1}, \{0\}_{k_1}\right)$, obtains bs_1,
2. $\mathcal{A}$ makes query $\mathcal{O}\left(\{0\}_{k_2}, \{0\}_{k_2}\right)$, obtains bs_2,
3. and so on ...
4. $\mathcal{A}$ makes query $\mathcal{O}(k_j, k_j)$ for every key in I,
5. $\mathcal{A}$ uses $\mathcal{O}$ to attack security of keys k_j for $j \notin I$.

Even problems close to adaptive selective decryption can be mapped.

In The Adaptive Model

Examples

We can map problems similar to selective decryption:

1. $\mathcal{A}$ makes query $\mathcal{O}\left(\{0\}_{k_1}, \{0\}_{k_1}\right)$, obtains bs_1,
2. $\mathcal{A}$ makes query $\mathcal{O}\left(\{0\}_{k_2}, \{0\}_{k_2}\right)$, obtains bs_2,
3. and so on ...
4. $\mathcal{A}$ makes query $\mathcal{O}(k_j, k_j)$ for every key in I,
5. $\mathcal{A}$ uses $\mathcal{O}$ to attack security of keys k_j for $j \notin I$.

Even problems close to adaptive selective decryption can be mapped.

In The Adaptive Model

Examples

We can map problems similar to selective decryption:

1. $\mathcal{A}$ makes query $\mathcal{O}\left(\{0\}_{k_1}, \{0\}_{k_1}\right)$, obtains bs_1,
2. $\mathcal{A}$ makes query $\mathcal{O}\left(\{0\}_{k_2}, \{0\}_{k_2}\right)$, obtains bs_2,
3. and so on ...
4. $\mathcal{A}$ makes query $\mathcal{O}(k_j, k_j)$ for every key in I,
5. $\mathcal{A}$ uses $\mathcal{O}$ to attack security of keys k_j for $j \notin I$.

Even problems close to adaptive selective decryption can be mapped.

In The Adaptive Model

Examples

We can map problems similar to selective decryption:

1. $\mathcal{A}$ makes query $\mathcal{O}\left(\{0\}_{k_1}, \{0\}_{k_1}\right)$, obtains bs_1,
2. $\mathcal{A}$ makes query $\mathcal{O}\left(\{0\}_{k_2}, \{0\}_{k_2}\right)$, obtains bs_2,
3. and so on ...
4. $\mathcal{A}$ makes query $\mathcal{O}(k_j, k_j)$ for every key in I,
5. $\mathcal{A}$ uses $\mathcal{O}$ to attack security of keys k_j for $j \notin I$.

Even problems close to adaptive selective decryption can be mapped.

In The Adaptive Model

Examples

We can map problems similar to selective decryption:

1. $\mathcal{A}$ makes query $\mathcal{O}\left(\{0\}_{k_1}, \{0\}_{k_1}\right)$, obtains bs_1,
2. $\mathcal{A}$ makes query $\mathcal{O}\left(\{0\}_{k_2}, \{0\}_{k_2}\right)$, obtains bs_2,
3. and so on ...
4. $\mathcal{A}$ makes query $\mathcal{O}(k_j, k_j)$ for every key in I,
5. $\mathcal{A}$ uses $\mathcal{O}$ to attack security of keys k_j for $j \notin I$.

Even problems close to adaptive selective decryption can be mapped.

Solving Selective Decryption

Weakening of the send-then-use hypothesis

If a key k is corrupted then in previous messages,

- k has not been used to encrypt other keys – no key emission.
- k has not been used to encrypt any encryption –no nested encryptions.

Theorem

Let $\mathcal{A}$ be a legal adversary which exec. time is bounded by p,
Let $u(p)$ be the best possible adv ag IND-CPA for Π in time p,

$$\mathbf{Adv}^{ad}_{\mathcal{A},\Pi} \leq pu(p)$$

The same technique solves selective decryption and adaptive selective decryption.

Solving Selective Decryption

Weakening of the send-then-use hypothesis

If a key k is corrupted then in previous messages,

- k has not been used to encrypt other keys – no key emission.
- k has not been used to encrypt any encryption –no nested encryptions.

Theorem

Let $\mathcal{A}$ be a legal adversary which exec. time is bounded by p,
Let $u(p)$ be the best possible adv ag IND-CPA for Π in time p,

$$\mathbf{Adv}^{ad}_{\mathcal{A},\Pi} \leq pu(p)$$

The same technique solves selective decryption and adaptive selective decryption.

Solving Selective Decryption

Weakening of the send-then-use hypothesis

If a key k is corrupted then in previous messages,

- k has not been used to encrypt other keys – no key emission.
- k has not been used to encrypt any encryption –no nested encryptions.

Theorem

Let $\mathcal{A}$ be a legal adversary which exec. time is bounded by p,
Let $u(p)$ be the best possible adv ag IND-CPA for Π in time p,

$$\mathbf{Adv}^{ad}_{\mathcal{A},\Pi} \leq pu(p)$$

The same technique solves selective decryption and adaptive selective decryption.

Nested Encryptions and Key Transmission

Weakening of the send-then-use hypothesis

If a key k is corrupted then in previous messages,

- k can be used to encrypt other keys.
- k has only been used to encrypt l nested encryptions: bounded nested encryptions.

$$\left\{\{\ldots\{0\}_{k_1}\ldots\}_{k_l}\right\}_k \text{ then } k \text{ can still be corrupted}$$

Theorem

Let $\mathcal{A}$ be a legal adversary which exec. time is bounded by p,

$$\mathbf{Adv}^{ad}_{\mathcal{A},\Pi} \leq \frac{(3p+1)^l - 1}{3} u(p)$$

Nested Encryptions and Key Transmission

Weakening of the send-then-use hypothesis

If a key k is corrupted then in previous messages,

- k can be used to encrypt other keys.
- k has only been used to encrypt l nested encryptions: bounded nested encryptions.

$$\left\{\{\ldots\{0\}_{k_1}\ldots\}_{k_l}\right\}_k \text{ then } k \text{ can still be corrupted}$$

Theorem

Let $\mathcal{A}$ be a legal adversary which exec. time is bounded by p,

$$\mathbf{Adv}^{ad}_{\mathcal{A},\Pi} \leq \frac{(3p+1)^l - 1}{3} u(p)$$

Conclusion and Future Work

Results

- Adaptive security of symbolic encryption with dynamic corruptions.
- But bounded number of encryption layers
- Can be applied to the Active setting.

Current and Future Works

1. Use static equivalence instead of patterns for symbolic equivalence for other primitives.
2. Application to protocol security.
3. Generalize the Adaptive and Active setting.

Conclusion and Future Work

Results

- Adaptive security of symbolic encryption with dynamic corruptions.
- But bounded number of encryption layers
- Can be applied to the Active setting.

Current and Future Works

1. Use static equivalence instead of patterns for symbolic equivalence for other primitives.
2. Application to protocol security.
3. Generalize the Adaptive and Active setting.