Computational soundness of observational equivalence

Hubert Comon-Lundh[1] and Véronique Cortier[2]

Spring school Cosyproof 2009, April 6th, 2009

[1]RCIS, AIST, Japan and ENS Cachan, France
[2]LORIA, CNRS, France

Two approaches

	Formal approach	Cryptographic approach
Messages	terms	bitstrings
Encryption	idealized	algorithm
Adversary	idealized	any polynomial algorithm
Secrecy property	reachability-based property	indistinguishability
Guarantees	unclear	strong

Two approaches

	Formal approach	Cryptographic approach
Messages	terms	bitstrings
Encryption	idealized	algorithm
Adversary	idealized	any polynomial algorithm
Secrecy property	reachability-based property	indistinguishability
Guarantees	unclear	strong
Proof	automatic	by hand and error-prone

Goal : Proving properties at the bitstring level using existing symbolic models.

Some related work

- Abadi-Rogaway (passive attackers)

$$[M_1, \ldots, M_k] \sim [M'_1, \ldots, M'_k] \Rightarrow [\![M_1, \ldots, M_k]\!] \approx [\![M'_1, \ldots, M'_k]\!]$$

- Backes-Pfitzman et al (active attackers)
  Simulatable cryptographic library

- Canetti-Herzog (active attackers)
  Universally composable symbolic analysis

- Warinschi et al (active attackers)
  Any concrete execution is captured by a symbolic execution (except with negligible probability).

Some related work

- Abadi-Rogaway (passive attackers)

$$[M_1, \ldots, M_k] \sim [M'_1, \ldots, M'_k] \Rightarrow [\![M_1, \ldots, M_k]\!] \approx [\![M'_1, \ldots, M'_k]\!]$$

- Backes-Pfitzman et al (active attackers)
  Simulatable cryptographic library
  $\rightarrow$ Mainly dedicated to trace properties + key secrecy
- Canetti-Herzog (active attackers)
  Universally composable symbolic analysis
  $\rightarrow$ Mainly dedicated to trace properties + key exchange
- Warinschi et al (active attackers)
  Any concrete execution is captured by a symbolic execution (except with negligible probability).
  $\rightarrow$ Mainly dedicated to trace properties + nonce secrecy

Trace properties vs observational equivalence

Fact 1 : Computational security properties are often stated as indistinguishability games rather than trace properties.
Example : secrecy, ideal functionalities, ...

Trace properties vs observational equivalence

Fact 1 : Computational security properties are often stated as indistinguishability games rather than trace properties.
Example : secrecy, ideal functionalities, ...

Fact 2 : Some security properties cannot be expressed as trace properties.
Example : Privacy properties of e-voting protocols

$$P(A,a)\|P(B,b) \sim_o P(A,b)\|P(B,a)$$

Indistinguishability

Definition (Computational indistinguishability)

$P \approx Q$ if for any adversary $\mathcal{A}$ (that is any PPT Turing machine)
$|\Pr\{r, r'(P(r) \| \mathcal{A}(r')) = 1\}| - |\Pr\{r, r'(Q(r) \| \mathcal{A}(r')) = 1\}|$
is negligible.

Intuitively, an attacker cannot tell the difference between P and Q.

Indistinguishability

Definition (Computational indistinguishability)

$P \approx Q$ if for any adversary $\mathcal{A}$ (that is any PPT Turing machine) $|\Pr\{r, r'(P(r) \| \mathcal{A}(r')) = 1\}| - |\Pr\{r, r'(Q(r) \| \mathcal{A}(r')) = 1\}|$ is negligible.

Intuitively, an attacker cannot tell the difference between P and Q.

There exists a similar symbolic definition !

Definition (observational equivalence)

$P \sim_o Q$ if for any process O, we have $P \| O \sim Q \| O$.

Intuitively, an observer cannot tell the difference between P and Q.

Our main result in brief

Observational equivalence is a sound abstraction of computational indistinguishability.

$$P \sim_o Q \Rightarrow [\![P]\!] \approx [\![Q]\!]$$

- For simple processes
  (A fragment of applied pi-calculus that captures most security protocols)
- For symmetric encryption implemented using IND-CC2 schemes

Outline of the talk

1. Setting
2. Soundness result
3. Proof sketch
4. Specific problems of symmetric encryption

Syntax (1)

- Terms with explicit destructors

$$
\begin{array}{rcll}
T & ::= & & \text{term of sort } s \\
 & | & x & \text{variable } x \text{ of sort s} \\
 & | & a & \text{name } a \text{ of sort s} \\
 & | & f(T_1, \ldots, T_k) & \text{application of symbol } f \in \mathcal{F}
\end{array}
$$

$\mathcal{F} = \{\mathrm{enc}, \mathrm{dec}, \langle \_, \_ \rangle, \pi_1, \pi_2\}$
$+$ concrete implementation $[\![T]\!]$: cryptographic encryption, decryption, pairing and projection functions

- Equational theory for pairing and symmetric encryption

$$\mathrm{dec}(\mathrm{enc}(x, y), y) = x, \quad \pi_1(\langle x, y \rangle) = x, \quad \pi_2(\langle x, y \rangle) = y$$

Syntax (2)

Predicates

- $M(s)$ holds whenever $s\downarrow$ contains no decryption nor projection symbols.
- $\mathrm{Eq}(s,t)$ holds whenever $M(s)$ and $M(t)$ hold and $s\downarrow = t\downarrow$
- $\mathrm{P}_{\mathrm{samekey}}$ is binary and holds on ciphertexts using the same encryption key.
- $EL(s,t)$ is binary and holds on terms on the same length.

Syntax (2)

Predicates

- $M(s)$ holds whenever $s\downarrow$ contains no decryption nor projection symbols.
- $\mathrm{Eq}(s,t)$ holds whenever $M(s)$ and $M(t)$ hold and $s\downarrow = t\downarrow$
- $\mathrm{P}_{\mathrm{samekey}}$ is binary and holds on ciphertexts using the same encryption key.
- $EL(s,t)$ is binary and holds on terms on the same length.

Two sequences of messages are statically equivalent, $\phi_1 \sim \phi_2$ if they satisfy the same predicates.

$$\phi_1 \models p(s_1,\ldots,s_k) \Leftrightarrow \phi_2 \models p(s_1,\ldots,s_k).$$

Intuitively, this should correspond to the ability of a computational adversary to distinguish between sequences of messages.

Basic processes

Role can be expressed through basic processes.

$$
\begin{aligned}
B := \quad & \mathbf{0} \\
& c(i_B, x).B \\
& \texttt{if } \phi \texttt{ then } \overline{c}(i_B, T).B \texttt{ else } \overline{c}(\perp)
\end{aligned}
$$

i_B : identifying name associated to the role (like e.g. an ip address)
Ensures that the intruder knows to who (s)he is talking to.

Simple processes

Simple processes = a fragment of the Applied pi-calculus [Abadi & Fournet].

$$(\nu k_1, \ldots, k_l) \quad (\nu n_1) B_1 \| \cdots \| (\nu n_k) B_k \| \; !(\nu n'_1) B'_1 \| \cdots \| \; !(\nu n'_p) B'_p$$

where the B_i, B'_i are basic processes.
This enforces in particular all communications to go through the attacker.

Remark : Each role is used for a bounded or an unbounded number of sessions.

Simple processes

Simple processes = a fragment of the Applied pi-calculus [Abadi & Fournet].

$$(\nu k_1, \ldots, k_l) \quad (\nu n_1)B_1 \| \cdots \| (\nu n_k)B_k \| \; !(\nu n'_1)B'_1 \| \cdots \| \; !(\nu n'_p)B'_p$$

where the B_i, B'_i are basic processes.
This enforces in particular all communications to go through the attacker.

Remark : Each role is used for a bounded or an unbounded number of sessions.

We also define the computational implementation $[\![P]\!]$ of a basic process P as expected.

Semantics : internal reduction

Internal reduction $\rightarrow$: mainly defined by the communication rule :

$$\overline{c}(M).P \parallel c(x).Q \quad \rightarrow \quad P \parallel Q\{x \mapsto M\} \ \mid \{x \mapsto M\}$$

Since communications are public

Semantics : internal reduction

Internal reduction $\rightarrow$: mainly defined by the communication rule :

$$\overline{c}(M).P \parallel c(x).Q \quad \rightarrow \quad P \parallel Q\{x \mapsto M\} \ \mid \{x \mapsto M\}$$

Since communications are public

Example :

$$\nu s, k.(\overline{c_1}(enc(s,k)) \| c_1(y).\overline{c_2}(dec(y,k)))$$
$$\rightarrow \ \nu s, k.\overline{c_2}(s) \mid \{y \mapsto enc(s,k)\}$$

$\{y \mapsto enc(s,k)\}$ is the active frame of process $\nu s, k.\overline{c_2}(s) \mid \{y \mapsto enc(s,k)\}$.

Observational equivalence

Two processes $\phi(P)$ and $\phi(Q)$ are observationally bisimilar if (informally) :

1. The processes $\phi(P)$ and $\phi(Q)$ can emit on the same channels ;
2. Any move $P \xrightarrow{\tau} P'$ can be matched by a move $Q \Longrightarrow Q'$.

such that $\phi(P)'$ and $\phi(Q)'$ remain observationally bisimilar (and reciprocally).

Observational equivalence

Two processes $\phi(P)$ and $\phi(Q)$ are observationally bisimilar if (informally) :

1. The processes $\phi(P)$ and $\phi(Q)$ can emit on the same channels ;
2. Any move $P \xrightarrow{\tau} P'$ can be matched by a move $Q \Longrightarrow Q'$.

such that $\phi(P)'$ and $\phi(Q)'$ remain observationally bisimilar (and reciprocally).

Definition

Two processes P et Q are *observational equivalent*, denoted $P \sim_o Q$, if for any process R, we have $P|R \sim Q|R$.

Soundness of observational equivalence

Theorem

For any simple processes P and Q

$$P \sim_o Q \implies [\![P]\!] \approx [\![Q]\!]$$

Applications :

- symbolic proof of privacy-like properties
- symbolic proof of anonymity
- symbolic proof of simulatability
- symbolic proof of secrecy (to some extend)
- ...

Hypotheses on the Implementation

- encryption : IND-CCA2 symmetric encryption scheme.
  $\longrightarrow$ the adversary cannot distinguish between $\{n_0\}_k$ and $\{n_1\}_k$ even if he has access to n_0 and n_1 and to encryption and decryption oracles.
- key hierarchy : there exists an order $<$ such that no key encrypts a smaller key.
- parsing :
  - each bit-string has a label which indicates his type (identity, nonce, key, ciphertext, ...)
  - ciphertext are tagged with a label that indicates which key is used.
    Typically $k = k_1 \| k_2$ and $\mathrm{enc}(m, k) = k_1 \| \{m\}_{k_2}$.

Hypotheses on the Implementation

- encryption : IND-CCA2 symmetric encryption scheme.
  $\longrightarrow$ the adversary cannot distinguish between $\{n_0\}_k$ and $\{n_1\}_k$ even if he has access to n_0 and n_1 and to encryption and decryption oracles.
- key hierarchy : there exists an order $<$ such that no key encrypts a smaller key.
- parsing :
  - each bit-string has a label which indicates his type (identity, nonce, key, ciphertext, ...)
  - ciphertext are tagged with a label that indicates which key is used.
    Typically $k = k_1 \| k_2$ and $\mathrm{enc}(m, k) = k_1 \| \{m\}_{k_2}$.
- authenticated key : the adversary can only use honestly generated keys (counter-examples otherwise).

Proof sketch

1. Mapping lemma for symmetric encryption and pairing.

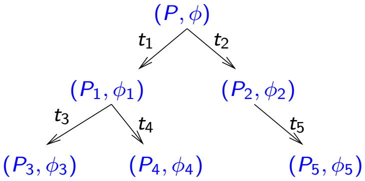
Theorem (mapping lemma)

Every concrete trace is the image of a valid formal trace, except with negligible probability, for symmetric encryption and pairing.

2. Introduction of process computation trees = generalized execution trees T_P.

$$P \sim_o Q \Rightarrow T_P \sim T_Q \Rightarrow T_P \approx T_Q \Rightarrow [\![P]\!] \approx [\![Q]\!]$$

$P \sim_o Q \Rightarrow T_P \sim T_Q$



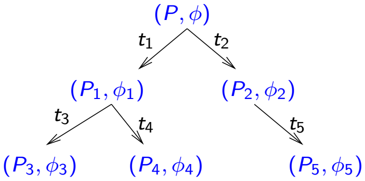
Process computation trees

Nodes are of the form (P, ϕ)

- P represents the current state of the protocol
- ϕ represents the messages already sent over the network

Arrows represent transitions

$P \sim_o Q \Rightarrow T_P \sim T_Q$



Process computation trees

Nodes are of the form (P, ϕ)

- P represents the current state of the protocol
- ϕ represents the messages already sent over the network

Arrows represent transitions

Definition : $T_1 \sim T_2$ if

- the root trees are in static equivalence
- there is a one-to-one mapping between sons of T_1 and sons of T_2 such that they are in equivalence.

Lemma : $P \sim_o Q \Rightarrow T_P \sim T_Q$

$T_P \sim T_Q \Rightarrow T_P \approx T_Q$

We associate to each computation tree T an oracle $\mathcal{O}_T$ that answers adversary's requests according to the tree T.
Note that initially, $[\![P]\!]$ has the same behavior than $\mathcal{O}_{T_P}$

$T_P \sim T_Q \Rightarrow T_P \approx T_Q$

We associate to each computation tree T an oracle $\mathcal{O}_T$ that answers adversary's requests according to the tree T.
Note that initially, $[\![P]\!]$ has the same behavior than $\mathcal{O}_{T_P}$

1. By IND-CCA2 security), $T \approx \Psi(T)$ where Ψ replaces honest encryption by encryption of zeros of the same length.
   Moreover, we can also show $T \sim \Psi(T)$

$T_P \sim T_Q \Rightarrow T_P \approx T_Q$

We associate to each computation tree T an oracle $\mathcal{O}_T$ that answers adversary's requests according to the tree T.
Note that initially, $[\![P]\!]$ has the same behavior than $\mathcal{O}_{T_P}$

1. By IND-CCA2 security), $T \approx \Psi(T)$ where Ψ replaces honest encryption by encryption of zeros of the same length.
   Moreover, we can also show $T \sim \Psi(T)$
2. We can check that $\Psi(T_1) \sim \Psi(T_2) \Rightarrow \Psi(T_1) = \Psi(T_2)$

$T_P \sim T_Q \Rightarrow T_P \approx T_Q$

We associate to each computation tree T an oracle $\mathcal{O}_T$ that answers adversary's requests according to the tree T.
Note that initially, $[\![P]\!]$ has the same behavior than $\mathcal{O}_{T_P}$

1. By IND-CCA2 security), $T \approx \Psi(T)$ where Ψ replaces honest encryption by encryption of zeros of the same length.
   Moreover, we can also show $T \sim \Psi(T)$
2. We can check that $\Psi(T_1) \sim \Psi(T_2) \Rightarrow \Psi(T_1) = \Psi(T_2)$
3. We deduce that

$$\begin{array}{llll} T_1 \sim T_2 & \Rightarrow \Psi(T_1) \sim \Psi(T_2) & \text{since} & T_i \sim \Psi(T_i) \\ & \Rightarrow \Psi(T_1) = \Psi(T_2) & & \\ & \Rightarrow T_1 \approx T_2 & \text{since} & T_i \approx \Psi(T_i) \end{array}$$

$T_P \approx T_Q \Rightarrow [\![P]\!] \approx [\![Q]\!]$

Theorem (mapping lemma)

Every concrete trace is the image of a valid formal trace, except with negligible probability, for symmetric encryption and pairing.

It means that any concrete trace of $[\![P]\!]$ interacting an adversary is the image of a trace in T_P. Thus

$$T_P \approx T_Q \Rightarrow [\![P]\!] \approx [\![Q]\!]$$

Hypothesis on dishonestly generated keys

In this work, we assume that dishonest keys are generated using the key generation scheme.

It would much more satisfactory to allow freely computed dishonest keys.

$\longrightarrow$ We provide pathological examples.

Decrypting with dishonest keys

$$
\begin{array}{llcll}
A \rightarrow & B & : & c & \text{c ciphertext} \\
B \rightarrow & A & : & N_b, \{N_b, c\}_{K_{ab}} & \\
A \rightarrow & B & : & k, \{N_b, c\}_{K_{ab}} & A \text{ releases her decryption key.} \\
B \rightarrow & & : & \text{bad state} & \text{if } B \text{ receives } k, \{N_b, \{N_b\}_k\}_{K_{ab}}
\end{array}
$$

Decrypting with dishonest keys

$$
\begin{array}{rcll}
A \rightarrow & B & : & c \\
B \rightarrow & A & : & N_b, \{N_b, c\}_{K_{ab}} \\
I \rightarrow & B & : & k, \{N_b, c\}_{K_{ab}} \\
B \rightarrow & & : & \text{bad state}
\end{array}
$$

c ciphertext

if B receives $k, \{N_b, \{N_b\}_k\}_{K_{ab}}$

Computational attack

The attacker can choose k such that $\mathrm{dec}(c, k) = N_b$.

Why it is possible

Security of encryption says nothing on dishonest keys !

Why it is possible

Security of encryption says nothing on dishonest keys !

Consider an (authenticated) IND-CCA2 scheme (G, E, D).

Consider the following authenticated) IND-CCA2 scheme (G', E', D') :

- Key generation $G' = 0.G$
- Decryption $D'(c, k)$
  - if $k = 0.k$ then output $D(c, k)$
  - if $k = 1.k$ then output k.
  - E' as E.

Why it is possible

Security of encryption says nothing on dishonest keys !

Consider an (authenticated) IND-CCA2 scheme (G, E, D).

Consider the following authenticated) IND-CCA2 scheme (G', E', D') :

- Key generation $G' = 0.G$
- Decryption $D'(c, k)$
  - if $k = 0.k$ then output $D(c, k)$
  - if $k = 1.k$ then output k.
  - E' as E.

In our previous example, the attacker can symply choose $k = N_b$.

Why it is possible

Security of encryption says nothing on dishonest keys !

Consider an (authenticated) IND-CCA2 scheme (G, E, D).

Consider the following authenticated) IND-CCA2 scheme (G', E', D') :

- Key generation $G' = 0.G$
- Decryption $D'(c, k)$
    - if $k = 0.k$ then output $D(c, k)$
    - if $k = 1.k$ then output k.
    - E' as E.

In our previous example, the attacker can symply choose $k = N_b$.

$\rightarrow$ Idea : enrich the symbolic setting (suggested by M. Backes)

E.g. $\dfrac{c \quad m}{\text{fakekey}(c, m)}$ $\qquad \text{dec}(c, \text{fakekey}(c, m)) = m$

Why it is possible

Security of encryption says nothing on dishonest keys !

Consider an (authenticated) IND-CCA2 scheme (G, E, D).

Consider the following authenticated) IND-CCA2 scheme (G', E', D') :

- Key generation $G' = 0.G$
- Decryption $D'(c, k)$
  - if $k = 0.k$ then output $D(c, k)$
  - if $k = 1.k$ then output k.
  - E' as E.

In our previous example, the attacker can symply choose $k = N_b$.

$\longrightarrow$ Idea : enrich the symbolic setting (suggested by M. Backes)

E.g. $$\frac{c \quad m}{\mathrm{fakekey}(c, m)} \qquad \mathrm{dec}(c, \mathrm{fakekey}(c, m)) = m$$

Does not work either.

Hidden ciphertext

$$
\begin{array}{llcll}
A \rightarrow & B & : & A, k, \{\{k'\}_k\}_{K_{ab}} & k, k' \text{ fresh keys} \\
B \rightarrow & A & : & \{k'\}_{K_{ab}} & \\
A \rightarrow & & : & \text{bad state} & \text{if } A \text{ receives } \{A\}_{K_{ab}}
\end{array}
$$

Hidden ciphertext

$$\begin{array}{llll} A \rightarrow & B & : & A, k, \{\{k'\}_k\}_{K_{ab}} \quad k, k' \text{ fresh keys} \\ B \rightarrow & A & : & \{k'\}_{K_{ab}} \\ A \rightarrow & & : & \text{bad state} \qquad \text{if } A \text{ receives } \{A\}_{K_{ab}} \end{array}$$

Computational attack

The attacker can choose k'' such that $\mathrm{dec}(\{k'\}_k, k'') = A$, even not knowing $\{k'\}_k$.

$$\begin{array}{llll} I \rightarrow & B & : & A, k'', \{\{k'\}_k\}_{K_{ab}} \\ B \rightarrow & A & : & \{A\}_{K_{ab}} \\ A \rightarrow & & : & \text{bad state!} \end{array}$$

Hidden ciphertext

$$\begin{array}{llll} A \rightarrow & B & : & A, k, \{\{k'\}_k\}_{K_{ab}} \quad k, k' \text{ fresh keys} \\ B \rightarrow & A & : & \{k'\}_{K_{ab}} \\ A \rightarrow & & : & \text{bad state} \quad \text{if } A \text{ receives } \{A\}_{K_{ab}} \end{array}$$

Computational attack

The attacker can choose k'' such that $\mathrm{dec}(\{k'\}_k, k'') = A$, even not knowing $\{k'\}_k$.

$\rightarrow$ idea : enrich again the symbolic setting ?

E.g. $\dfrac{m}{\mathrm{fakekey2}(m)}$ $\quad \mathrm{dec}(c, \mathrm{fakekey2}(m)) = m$ for any c

Simultaneous ciphertexts

$$
\begin{array}{llcl}
A \rightarrow & B & : & c_1, \ldots, c_p \qquad\qquad c_1, \ldots, c_p \text{ ciphertexts} \\
B \rightarrow & A & : & \{N_b, c_1, \ldots, c_p\}_{K_{ab}}, N_1, \ldots, N_p \\
A \rightarrow & B & : & k, \{N_b, c_1, \ldots, c_p\}_{K_{ab}} \\
B \rightarrow & & : & \text{bad state} \quad \text{if } B \text{ receives } k, \{N_b, \{N_1\}_k, \ldots, \{N_p\}_k\}_{K_{ab}}
\end{array}
$$

Simultaneous ciphertexts

$$
\begin{array}{rlcl}
I \rightarrow & B & : & c_1, \ldots, c_p \qquad\qquad\qquad c_1, \ldots, c_p \text{ ciphertexts} \\
B \rightarrow & A & : & \{N_b, c_1, \ldots, c_p\}_{K_{ab}}, N_1, \ldots, N_p \\
I \rightarrow & B & : & k', \{N_b, c_1, \ldots, c_p\}_{K_{ab}} \\
B \rightarrow & & : & \text{bad state} \quad \text{if } B \text{ receives } k, \{N_b, \{N_1\}_k, \ldots, \{N_p\}_k\}_{K_{ab}}
\end{array}
$$

Computational attack

The attacker chooses $c_1, \ldots, c_p$ and k' such that $\mathrm{dec}(c_i, k') = N_b$ for all $1 \leq i \leq p$.

Simultaneous ciphertexts

$$\begin{array}{rcll} I \rightarrow & B & : & c_1, \ldots, c_p \qquad\qquad c_1, \ldots, c_p \text{ ciphertexts} \\ B \rightarrow & A & : & \{N_b, c_1, \ldots, c_p\}_{K_{ab}}, N_1, \ldots, N_p \\ I \rightarrow & B & : & k', \{N_b, c_1, \ldots, c_p\}_{K_{ab}} \\ B \rightarrow & & : & \text{bad state} \quad \text{if } B \text{ receives } k, \{N_b, \{N_1\}_k, \ldots, \{N_p\}_k\}_{K_{ab}} \end{array}$$

Computational attack

The attacker chooses $c_1, \ldots, c_p$ and k' such that $\mathrm{dec}(c_i, k') = N_b$ for all $1 \leq i \leq p$.

$\rightarrow$ idea : Yet another rule ?

$$\frac{c_1 \quad \cdots \quad c_p \quad m_1 \quad \cdots \quad m_p}{\mathrm{fakekey3}(c_1, \ldots, c_p, m_1, \ldots, m_p)}$$

$$\mathrm{dec}(c_i, \mathrm{fakekey3}(c_1, \ldots, c_p, m_1, \ldots, m_p)) = m_i$$

Playing with dishonest encryption

$$\begin{array}{llll} A \rightarrow & B & : & \{N_a\}_{K_{ab}} \\ C \rightarrow & B & : & k \\ B \rightarrow & A & : & k, \{\{N_a\}_k\}_{K_{ab}} \\ A \rightarrow & & : & \text{bad state} \end{array}$$

c ciphertext

if A receives $k, \{\{N_a, N_a\}_k\}_{K_{ab}}$

Playing with dishonest encryption

$$
\begin{array}{llll}
A \rightarrow B & : & \{N_a\}_{K_{ab}} & c \text{ ciphertext} \\
C \rightarrow B & : & k' & \\
B \rightarrow A & : & k', \{\{N_a\}_{k'}\}_{K_{ab}} & \\
A \rightarrow & : & \text{bad state} & \text{if } A \text{ receives } k, \{\{N_a, N_a\}_k\}_{K_{ab}}
\end{array}
$$

Computational attack

The attacker can choose k' such that $\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, N_a$

Playing with dishonest encryption

$$
\begin{array}{llcl}
A \rightarrow & B & : & \{N_a\}_{K_{ab}} \\
C \rightarrow & B & : & k' \\
B \rightarrow & A & : & k', \{\{N_a\}_{k'}\}_{K_{ab}} \\
A \rightarrow & & : & \text{bad state}
\end{array}
$$

c ciphertext

if A receives $k, \{\{N_a, N_a\}_k\}_{K_{ab}}$
$k, \{\{N_a, N_a, N_a\}_k\}_{K_{ab}}$

Computational attack

The attacker can choose k' such that $\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, N_a$
$\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, N_a, N_a$

Playing with dishonest encryption

$$\begin{array}{llcl} A \rightarrow & B & : & \{N_a\}_{K_{ab}} \\ C \rightarrow & B & : & k' \\ B \rightarrow & A & : & k', \{\{N_a\}_{k'}\}_{K_{ab}} \\ A \rightarrow & & : & \text{bad state} \end{array}$$

c ciphertext

if A receives $k, \{\{N_a, N_a\}_k\}_{K_{ab}}$
$k, \{\{N_a, N_a, N_a\}_k\}_{K_{ab}}$
$k, \{\{N_a, A\}_k\}_{K_{ab}}$
...

Computational attack

The attacker can choose k' such that $\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, N_a$
$\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, N_a, N_a$
$\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, A$
...

Playing with dishonest encryption

$$
\begin{array}{llll}
A \to & B & : & \{N_a\}_{K_{ab}} \\
C \to & B & : & k' \\
B \to & A & : & k', \{\{N_a\}_{k'}\}_{K_{ab}} \\
A \to & & : & \text{bad state}
\end{array}
$$

c ciphertext

if A receives $k, \{\{N_a, N_a\}_k\}_{K_{ab}}$
$k, \{\{N_a, N_a, N_a\}_k\}_{K_{ab}}$
$k, \{\{N_a, A\}_k\}_{K_{ab}}$
$\ldots$

Computational attack

The attacker can choose k' such that $\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, N_a$
$\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, N_a, N_a$
$\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, A$
$\ldots$

M. Backes current solution : For any cypher-text c, for any dishonestly generated key k, $\mathrm{dec}(c, k)$ may yield any term.

Conclusion

$$P \sim_o Q \Rightarrow [\![P]\!] \approx [\![Q]\!]$$

It is possible to use existing formal models to prove indistinguishability-based security properties at the bit-string level

Application : Automatic computationally sound proof using for example ProVerif

Conclusion

$$P \sim_o Q \Rightarrow [\![P]\!] \approx [\![Q]\!]$$

It is possible to use existing formal models to prove indistinguishability-based security properties at the bit-string level

Application : Automatic computationally sound proof using for example ProVerif

Further work :

- Extension to more cryptographic primitives : asymmetric encryption, signatures, macs, ...
- Composition result :
  trace mapping + soundness of static equivalence for adaptive adversaries $\Rightarrow$ soundness of observational equivalence ?
- Extension to security properties with synchronization phase