Deciding Indistinguishability: A Decision Result for a Set of Cryptographic Game Transformations

Adrien Koutsos

March 13, 2018
1 Introduction

2 The Model

3 Game Transformations
   - Basic Games
   - Game Transformations

4 Decision Result

5 Conclusion
Security protocols are distributed programs which aim at providing some security properties.

They are extensively used, and bugs can be very costly.

Security protocols are often short, but the security properties are complex.

⇒ Need to use formal methods.
Introduction

Goal of this work

We focus on fully automatic proofs of indistinguishability properties in the computational model:

- **Computational model:** the adversary is any *probabilistic polynomial time Turing machine*. This offers strong security guarantees.
- **Indistinguishability properties:** e.g. strong secrecy, anonymity or unlinkability.
- **Fully automatic:** we want a complete decision procedure.
The Private Authentication Protocol

\[
\begin{align*}
A' & : \quad n_{A'} \leftarrow \\
B & : \quad n_B \leftarrow \\
1: A' & \rightarrow B : \quad \{\langle pk(A'), n_{A'}\rangle\}_{pk(B)} \\
2: B & \rightarrow A' : \quad \begin{cases} 
\{\langle n_{A'}, n_B\rangle\}_{pk(A)} & \text{if } pk(A') = pk(A) \\
\{\langle n_B, n_B\rangle\}_{pk(A)} & \text{otherwise}
\end{cases}
\end{align*}
\]
1 Introduction

2 The Model

3 Game Transformations
   - Basic Games
   - Game Transformations

4 Decision Result

5 Conclusion
In the computational model, a message is a *distribution over bitstrings*. We only consider distribution built using:

- Random uniform sampling $n_A, n_B \ldots$ over $\{0, 1\}^\eta$.
- Function applications:
  - $A, B, \langle _, _ \rangle, \pi_i(_), \{ _ \}_, pk(_), sk(_), if _ then _ else _ \ldots$
In the computational model, a message is a *distribution over bitstrings*. We only consider distribution built using:

- Random uniform sampling $n_A, n_B \ldots$ over $\{0, 1\}^\eta$.
- Function applications:
  - $A, B, \langle \_, \_ \rangle, \pi_i(\_), \{\_\}_\_, pk(\_), sk(\_), if \_ then \_ else \_ \ldots$.

**Examples**

$$\langle n_A, A \rangle \quad \pi_1(n_B) \quad \{\langle pk(A'), n_{A'} \rangle\}_{pk(B)}$$
## The Private Authentication Protocol

1. $A' \rightarrow B : \{\langle \text{pk}(A') , n_{A'} \rangle \}_{\text{pk}(B)}$

2. $B \rightarrow A' : \begin{cases} \{\langle n_{A'} , n_B \rangle \}_{\text{pk}(A)} & \text{if } \text{pk}(A') = \text{pk}(A) \\ \{\langle n_B , n_B \rangle \}_{\text{pk}(A)} & \text{otherwise} \end{cases}$

### How do we represent the adversary’s inputs?

We use special functions symbols $g, g_0, g_1, \ldots$. Intuitively, they can be any probabilistic polynomial time algorithm. Moreover, branching of the protocol is done using if _ then _ else _.
The Private Authentication Protocol

1: $A' \rightarrow B : \{\langle pk(A') , n_{A'} \rangle \}_{pk(B)}$

2: $B \rightarrow A' : \begin{cases} \{\langle n_{A'}, n_B \rangle \}_{pk(A)} & \text{if } pk(A') = pk(A) \\ \{\langle n_B , n_B \rangle \}_{pk(A)} & \text{otherwise} \end{cases}$

How do we represent the adversary’s inputs?

- We use special functions symbols $g, g_0, g_1, \ldots$.
The Private Authentication Protocol

1 : A' → B : \{\langle pk(A'), n_{A'} \rangle \}_{pk(B)}

2 : B → A' : \begin{cases} \{\langle n_{A'}, n_B \rangle \}_{pk(A)} & \text{if } pk(A') = pk(A) \\ \{\langle n_B, n_B \rangle \}_{pk(A)} & \text{otherwise} \end{cases}

How do we represent the adversary’s inputs?

- We use special functions symbols \( g, g_0, g_1 \ldots \).
- Intuitively, they can be any \textit{probabilistic polynomial time algorithm}.
- Moreover, branching of the protocol is done using if \_ then \_ else \_.

Adrien Koutsos
Deciding Indistinguishability
March 13, 2018 8 / 37
The Private Authentication Protocol

1: $A' \rightarrow B : \{\langle \text{pk}(A'), n_{A'} \rangle \}_{\text{pk}(B)}$

2: $B \rightarrow A' : \begin{cases} \{\langle n_{A'}, n_B \rangle \}_{\text{pk}(A)} & \text{if } \text{pk}(A') = \text{pk}(A) \\ \{\langle n_B, n_B \rangle \}_{\text{pk}(A)} & \text{otherwise} \end{cases}$
Model: Messages

The Private Authentication Protocol

1: $A' \rightarrow B : \{\langle pk(A'), n_{A'} \rangle \}_{pk(B)}$

2: $B \rightarrow A' : \begin{cases} \{\langle n_{A'}, n_B \rangle \}_{pk(A)} & \text{if } pk(A') = pk(A) \\ \{\langle n_B, n_B \rangle \}_{pk(A)} & \text{otherwise} \end{cases}$

Term Representing the Messages in PA

$t_1 = \{\langle pk(A'), n_{A'} \rangle \}_{pk(B)}$

$t_2 = \begin{cases} EQ(\pi_1(\text{dec}(g(t_1), \text{sk}(B))); pk(A)) & \text{then } \{\langle \pi_2(\text{dec}(g(t_1), \text{sk}(B))), n_B \rangle \}_{pk(A)} \\ \{\langle n_B, n_B \rangle \}_{pk(A)} & \text{else} \end{cases}$
Protocol Execution

The execution of a protocol $P$ is a sequence of terms using adversarial function symbols:

$$u_0^P, \ldots, u_n^P$$

where $u_i^P$ is the $i$-th message sent on the network by $P$. 

Remark

Only possible for a bounded number of sessions. The sequence of terms can be automatically computed (folding).
Protocol Execution

The execution of a protocol $P$ is a sequence of terms using adversarial function symbols:

$$u_0^P, \ldots, u_n^P$$

where $u_i^P$ is the $i$-th message sent on the network by $P$.

Remark

- Only possible for a bounded number of sessions.
- The sequence of terms can be automatically computed (folding).
Two protocols $P$ and $Q$ are *indistinguishable* if every adversary $A$ loses the following game:

- We toss a coin $b$.
- If $b = 0$, then $A$ interacts with $P$. Otherwise $A$ interacts with $Q$.

*Remark:* $A$ is an active adversary (it is the network).

- After the protocol execution, $A$ outputs a guess $b'$ for $b$.

$A$ wins if it guesses correctly with probability better than $\approx 1/2$. 
Model: Security Properties

Proposition

$P$ and $Q$ are indistinguishable

$\iff$

$u_0^P, \ldots, u_n^P$ and $u_0^Q, \ldots, u_n^Q$ are indistinguishable

$\iff$

$u_0^P, \ldots, u_n^P \sim u_0^Q, \ldots, u_n^Q$
Proposition

\[ P \text{ and } Q \text{ are indistinguishable} \iff \]

\[ u_0^P, \ldots, u_n^P \text{ and } u_0^Q, \ldots, u_n^Q \text{ are indistinguishable} \iff \]

\[ u_0^P, \ldots, u_n^P \sim u_0^Q, \ldots, u_n^Q \]

Example: Privacy for PA

\[ t_1^A, t_2^A \sim t_1^A', t_2^A' \]
Messages are represented by terms, which are built using names $N$ and function symbols $\mathcal{F}$.

A protocol execution is represented by a sequence of terms.

Indistinguishability properties are expressed through games:

$$u_0^P, \ldots, u_n^P \sim u_0^Q, \ldots, u_n^Q$$
1 Introduction

2 The Model

3 Game Transformations
   - Basic Games
   - Game Transformations

4 Decision Result

5 Conclusion
We know that some indistinguishability games are secure:

- Using $\alpha$-renaming of random samplings:

$$n_A, n_B \sim n_C, n_D$$
We know that some indistinguishability games are secure:

- Using $\alpha$-renaming of random samplings:
  \[ n_A, n_B \sim n_C, n_D \]

- Using probabilistic arguments:
  
  when $n_A \notin \text{st}(t)$, \[ \begin{align*} 
  t \oplus n_A &\sim n_B \\
  \text{EQ}(t; n_A) &\sim \text{false} 
  \end{align*} \]
Basic Games

We know that some indistinguishability games are secure:

- Using $\alpha$-renaming of random samplings:

  \[ n_A, n_B \sim n_C, n_D \]

- Using probabilistic arguments:

  \[
  \begin{align*}
  t \oplus n_A &\sim n_B \\
  \text{EQ}(t; n_A) &\sim \text{false}
  \end{align*}
  \]
  \[ \text{when } n_A \notin \text{st}(t), \]

- Using cryptographic assumptions on the security primitives, e.g. if $\{ \_ \}, \text{dec}(\_ , \_), \text{pk}(\_), \text{sk}(\_)$ is $\text{IND-CCA}1$. 

Adrien Koutsos
Cryptographic assumptions: **IND-CCA1**

A

<table>
<thead>
<tr>
<th>pk</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
</tr>
<tr>
<td>( x_1 )</td>
</tr>
<tr>
<td>( \ldots )</td>
</tr>
<tr>
<td>( c_n )</td>
</tr>
<tr>
<td>( x_n )</td>
</tr>
<tr>
<td>( (m_0, m_1) )</td>
</tr>
<tr>
<td>( y )</td>
</tr>
<tr>
<td>( b' )</td>
</tr>
</tbody>
</table>

**Challenger**

\( b \leftarrow \{0, 1\} \);

\( (pk, sk) \leftarrow KG(1^\eta) \);

\( x_1 \leftarrow \text{dec}(c_1, sk) \);

\( x_n \leftarrow \text{dec}(c_n, sk) \);

\( y \leftarrow \{m_b\}_{pk} \);

\( b = b' \)?
Basic Game: Cryptographic Assumptions

$Enc_{\text{CCA1}}$ Games:

$\vec{v}, \{m_0\}_{pk} \sim \vec{v}, \{m_1\}_{pk}$
Basic Game: Cryptographic Assumptions

Enc_{CCA1} Games:

\[ \vec{v}, \{ m_0 \}_{pk} \sim \vec{v}, \{ m_1 \}_{pk} \]

Assuming:
- sk occurs only in decryption position in \( \vec{v}, m_0, m_1 \).

Theorem

The \( Enc_{CCA1} \) games are secure when the encryption and decryption function are an IND-CCA1 encryption scheme.
Basic Game: Cryptographic Assumptions

Enc_{CCA1} Games:

\[ \vec{v}, \{m_0\}_{pk} \sim \vec{v}, \{m_1\}_{pk} \]

Assuming:

- sk occurs only in decryption position in \( \vec{v}, m_0, m_1 \).

Theorem

The Enc_{CCA1} games are secure when the encryption and decryption function are an IND-CCA1 encryption scheme.

Other cryptographic assumptions

IND-CPA, IND-CCA2, CR, PRF, EUF-CMA ...
Game Transformations

Proof Technique

- If $\vec{u} \sim \vec{v}$ is not a basic game, we try to show that it is secure through a succession of *game transformations*:

  $$\begin{align*}
  \vec{s} & \sim \vec{t} \\
  \vec{u} & \sim \vec{v}
  \end{align*}$$

- This is the way cryptographers or CryptoVerif do proofs.
If $\vec{u} \sim \vec{v}$ is not a basic game, we try to show that it is secure through a succession of game transformations:

$$
\frac{\vec{s} \sim \vec{t}}{\vec{u} \sim \vec{v}}
$$

This is the way cryptographers or CryptoVerif do proofs.

Validity by reduction: $\vec{u} \sim \vec{v}$ can be replaced by $\vec{s} \sim \vec{t}$ when, given an adversary winning $\vec{u} \sim \vec{v}$, we can build an adversary winning $\vec{s} \sim \vec{t}$.
Game Transformations

Proof Technique

- If $\vec{u} \sim \vec{v}$ is not a basic game, we try to show that it is secure through a succession of game transformations:

\[
\frac{\vec{s} \sim \vec{t}}{\vec{u} \sim \vec{v}}
\]

- This is the way cryptographers or CryptoVerif do proofs.

- **Validity by reduction:** $\vec{u} \sim \vec{v}$ can be replaced by $\vec{s} \sim \vec{t}$ when, given an adversary winning $\vec{u} \sim \vec{v}$, we can build an adversary winning $\vec{s} \sim \vec{t}$.

Example

\[
\frac{x \sim y}{y \sim x} \text{ Sym}
\]
Duplicate

\[
\begin{array}{c}
x \sim y \\
x, x \sim y, y
\end{array}
\quad \text{Dup}
\]
Duplicate

\[ \vec{w}_l, x \sim \vec{w}_r, y \implies \vec{w}_l, x, x \sim \vec{w}_r, y, y \]

- Dup
If you cannot distinguish the arguments, you cannot distinguish the images.

\[
\begin{align*}
&x_1, \ldots, x_n \sim y_1, \ldots, y_n \\
&\frac{x_1, \ldots, x_n}{x_1, \ldots, x_n} \sim \frac{y_1, \ldots, y_n}{y_1, \ldots, y_n} \\
&f(x_1, \ldots, x_n) \sim f(y_1, \ldots, y_n) \quad \text{FA}
\end{align*}
\]
If you cannot distinguish the arguments, you cannot distinguish the images.

\[ \vec{w}_l, x_1, \ldots, x_n \sim \vec{w}_r, y_1, \ldots, y_n \]

\[ \vec{w}_l, f(x_1, \ldots, x_n) \sim \vec{w}_r, f(y_1, \ldots, y_n) \]

FA
Case Study

If we use Function Application on \((\text{if then else })\):

\[
\frac{b, u, v \sim b', u', v'}{\text{if } b \text{ then } u \text{ else } v \sim \text{if } b' \text{ then } u' \text{ else } v'} \quad \text{FA}
\]
If we use Function Application on \( (\text{if then else} ) \):

\[
\text{if } b \text{ then } u \text{ else } v \sim \text{if } b' \text{ then } u' \text{ else } v'
\]

But we can do better:

\[
\begin{align*}
\text{if } b \text{ then } u \text{ else } v & \sim \text{if } b' \text{ then } u' \text{ else } v' \\
\text{if } b' \text{ then } u' \text{ else } v' & \sim \text{if } b' \text{ then } u' \text{ else } v'
\end{align*}
\]
If we use Function Application on (if then else):

\[
\begin{align*}
& b, u, v \sim b', u', v' \\
& \text{if } b \text{ then } u \text{ else } v \sim \text{if } b' \text{ then } u' \text{ else } v' \quad \text{FA}
\end{align*}
\]

But we can do better:

\[
\begin{align*}
& \vec{w}_l, b, u \sim \vec{w}_r, b', u' \\
& \vec{w}_l, b, v \sim \vec{w}_r, b', v' \\
& \vec{w}_l, \text{if } b \text{ then } u \text{ else } v \sim \vec{w}_r, \text{if } b' \text{ then } u' \text{ else } v' \quad \text{CS}
\end{align*}
\]
Remark: $\sim$ is not a congruence!

**Counter-Example:** $n \sim n$ and $n \sim n'$, but $n, n \not\sim n, n'$. 
Remark: $\sim$ is not a congruence!

Counter-Example: $n \sim n$ and $n \sim n'$, but $n, n \not\sim n, n'$.

**Congruence**

If $\text{EQ}(u; v) \sim \text{true}$ then $u$ and $v$ are (almost always) equal

$\Rightarrow$ we have a congruence.

$u = v$ syntactic sugar for $\text{EQ}(u; v) \sim \text{true}$

**Equational Theory: Protocol Functions**

- $\pi_i(\langle x_1, x_2 \rangle) = x_i$
  
- $\text{dec}(\{x\}_pk(y), sk(y)) = x$

$i \in \{1, 2\}$
Equational Theory: Protocol Functions

If Homomorphism:

$$f(\vec{u}, \text{if } b \text{ then } x \text{ else } y, \vec{v}) = \text{if } b \text{ then } f(\vec{u}, x, \vec{v}) \text{ else } f(\vec{u}, y, \vec{v})$$

$$\text{if (if } b \text{ then } a \text{ else } c \text{) then } x \text{ else } y =$$

$$\text{if } b \text{ then (if } a \text{ then } x \text{ else } y) \text{ else (if } c \text{ then } x \text{ else } y)$$
Equational Theory: Protocol Functions

**If Homomorphism:**

\[
f(\vec{u}, \text{if } b \text{ then } x \text{ else } y, \vec{v}) = \text{if } b \text{ then } f(\vec{u}, x, \vec{v}) \text{ else } f(\vec{u}, y, \vec{v})
\]

\[
\text{if } (\text{if } b \text{ then } a \text{ else } c) \text{ then } x \text{ else } y = \\
\text{if } b \text{ then } (\text{if } a \text{ then } x \text{ else } y) \text{ else } (\text{if } c \text{ then } x \text{ else } y)
\]

**If Rewriting:**

\[
\text{if } b \text{ then } x \text{ else } x = x
\]

\[
\text{if } b \text{ then } (\text{if } b \text{ then } x \text{ else } y) \text{ else } z = \text{if } b \text{ then } x \text{ else } z
\]

\[
\text{if } b \text{ then } x \text{ else } (\text{if } b \text{ then } y \text{ else } z) = \text{if } b \text{ then } x \text{ else } z
\]
Game Transformation: Term Rewriting System

Equational Theory: Protocol Functions

**If Homomorphism:**
\[ f(\overline{u}, \text{if } b \text{ then } x \text{ else } y, \overline{v}) = \text{if } b \text{ then } f(\overline{u}, x, \overline{v}) \text{ else } f(\overline{u}, y, \overline{v}) \]
\[ \text{if (if } b \text{ then } a \text{ else } c \text{) then } x \text{ else } y = \]
\[ \quad \text{if } b \text{ then (if } a \text{ then } x \text{ else } y \text{) else (if } c \text{ then } x \text{ else } y \text{)} \]

**If Rewriting:**
\[ \text{if } b \text{ then } x \text{ else } x = x \]
\[ \text{if } b \text{ then (if } b \text{ then } x \text{ else } y \text{) else } z = \text{if } b \text{ then } x \text{ else } z \]
\[ \text{if } b \text{ then } x \text{ else (if } b \text{ then } y \text{ else } z \text{) = if } b \text{ then } x \text{ else } z \]

**If Re-Ordering:**
\[ \text{if } b \text{ then (if } a \text{ then } x \text{ else } y \text{) else } z = \]
\[ \quad \text{if } a \text{ then (if } b \text{ then } x \text{ else } z \text{) else (if } b \text{ then } y \text{ else } z \text{)} \]
\[ \text{if } b \text{ then } x \text{ else (if } a \text{ then } y \text{ else } z \text{) =} \]
\[ \quad \text{if } a \text{ then (if } b \text{ then } x \text{ else } y \text{) else (if } b \text{ then } x \text{ else } z \text{)} \]
1 Introduction

2 The Model

3 Game Transformations
   - Basic Games
   - Game Transformations

4 Decision Result

5 Conclusion
Decision Problem: Game Transformations

**Input:** A game $\vec{u} \sim \vec{v}$.

**Question:** Is there a sequence of game transformations in $Ax$ showing that $\vec{u} \sim \vec{v}$ is secure?
Decidability

Decision Problem: Game Transformations

**Input:** A game $\vec{u} \sim \vec{v}$.

**Question:** Is there a sequence of game transformations in $Ax$ showing that $\vec{u} \sim \vec{v}$ is secure?

or equivalently

Decision Problem: Satisfiability

**Input:** A ground formula $\vec{u} \sim \vec{v}$ in the BC indistinguishability logic.

**Question:** Is $Ax \land \vec{u} \not\sim \vec{v}$ satisfiable?
The Non-Basic Game Transformations in Ax

\[
\frac{x \sim y}{x, x \sim y, y} \text{ Dup}
\]

\[
\frac{x_1, \ldots, x_n \sim y_1, \ldots, y_n}{f(x_1, \ldots, x_n) \sim f(y_1, \ldots, y_n)} \text{ FA}
\]

\[
\frac{b, u \sim b', u' \quad b, v \sim b', v'}{\text{if } b \text{ then } u \text{ else } v \sim \text{if } b' \text{ then } u' \text{ else } v'} \text{ CS}
\]
The Non-Basic Game Transformations in \([\text{Ax}]
\)

- **Dup**
  \[
  \frac{x \sim y}{x, x \sim y, y}
  \]

- **FA**
  \[
  \frac{x_1, \ldots, x_n \sim y_1, \ldots, y_n}{f(x_1, \ldots, x_n) \sim f(y_1, \ldots, y_n)}
  \]

- **CS**
  \[
  \frac{b, u \sim b', u'}{\text{if } b \text{ then } u \text{ else } v \sim \text{if } b' \text{ then } u' \text{ else } v'}
  \]

- **R**
  \[
  \frac{\vec{u}' \sim \vec{v}'}{\vec{u} \sim \vec{v}}
  \]

when \(\vec{u} \equiv_R \vec{u}'\) and \(\vec{v} \equiv_R \vec{v}'\)
Theorem

There exists a term rewriting system $\rightarrow_R \subseteq =$ such that:

- $\rightarrow_R$ is convergent.
- $=$ is equal to $(\leftarrow R \cup \rightarrow_R)^*$. 
Deconstructing Rules

Rules CS, FA and Dup are decreasing transformations.
Strategy

Deconstructing Rules
Rules CS, FA and Dup are decreasing transformations.

Problems
- The rule $R$ is not decreasing!
- The basic games (CCA1) are given through a recursive schema.
Strategy

Deconstructing Rules
Rules CS, FA and Dup are decreasing transformations.

Problems
- The rule $R$ is not decreasing!
- The basic games (CCA1) are given through a recursive schema.

Naive Idea
$R$ is convergent, so could we restrict proofs to terms in $R$-normal form?
Difficulties

If Introduction: $x \rightarrow \text{if } b \text{ then } x \text{ else } x$

\[ n \sim \text{if } g() \text{ then } n \text{ else } n' \]
Difficulties

If Introduction: \( x \rightarrow \text{if } b \text{ then } x \text{ else } x \)

\[
\begin{align*}
\text{if } g() \text{ then } n \text{ else } n & \sim \text{if } g() \text{ then } n \text{ else } n' \\
& \text{R} \\
& n \sim \text{if } g() \text{ then } n \text{ else } n'
\end{align*}
\]
Difficulties

If Introduction: $x \rightarrow \text{if } b \text{ then } x \text{ else } x$

\[
\frac{n \sim n}{g(), n \sim g(), n} \quad \text{FA} \quad \frac{n \sim n'}{g(), n \sim g(), n'} \quad \text{FA}
\]

\[
\frac{\text{if } g() \text{ then } n \text{ else } n \sim \text{if } g() \text{ then } n \text{ else } n'}{n \sim \text{if } g() \text{ then } n \text{ else } n'} \quad \text{CS} \quad \text{R}
\]
Difficulties

If Introduction: \[ x \rightarrow \text{if } b \text{ then } x \text{ else } x \]

\[ u, n \sim u, \text{if } g(u) \text{ then } n \text{ else } n' \]
Difficulties

If Introduction: $\forall x \rightarrow \text{if } b \text{ then } x \text{ else } x$

\[
\begin{array}{c}
\vec{u}, \text{if } g(\vec{u}) \text{ then } n \text{ else } n \sim \vec{u}, \text{if } g(\vec{u'}) \text{ then } n \text{ else } n' \\
\vec{u}, n \sim \vec{u}, \text{if } g(\vec{u'}) \text{ then } n \text{ else } n'
\end{array}
\]
Difficulties

If Introduction: \( x \rightarrow \text{if } b \text{ then } x \text{ else } x \)

\[
\begin{align*}
\vec{u}, n & \sim \vec{u}, n \quad \text{FA, Dup} \\
\vec{u}, g(\vec{u}), n & \sim \vec{u}, g(\vec{u}), n \quad \text{FA, Dup} \\
\vec{u}, n & \sim \vec{u}, n' \quad \text{CS} \\
\vec{u}, g(\vec{u}), n & \sim \vec{u}, g(\vec{u}), n' \quad \text{R}
\end{align*}
\]
Difficulties

If Introduction: \( x \rightarrow \text{if } b \text{ then } x \text{ else } x \)

\[
\frac{\vec{u}, n \sim \vec{u}, n}{\vec{u}, g(\vec{u}), n \sim \vec{u}, g(\vec{u}), n} \quad \text{FA, Dup}
\]

\[
\frac{\vec{u}, n \sim \vec{u}, n'}{\vec{u}, g(\vec{u}), n \sim \vec{u}, g(\vec{u}), n'} \quad \text{FA, Dup}
\]

\[
\vec{u}, \text{if } g(\vec{u}) \text{ then } n \text{ else } n' \sim \vec{u}, \text{if } g(\vec{u}) \text{ then } n \text{ else } n'
\]

\[
\vec{u}, n \sim \vec{u}, \text{if } g(\vec{u}) \text{ then } n \text{ else } n'
\]

Bounded Introduction

Still, the introduced conditional \( g(\vec{u}) \) is bounded by the other side.
Proof Cut: Introduction of a Conditional on Both Sides

\[ \frac{a, s \sim b, t}{\text{if } a \text{ then } s \text{ else } s \sim \text{ if } b \text{ then } t \text{ else } t} \]

\[ s \sim t \]

\[ \text{CS R} \]
Proof Cut: Introduction of a Conditional on Both Sides

\[
\frac{a, s \sim b, t}{\text{if } a \text{ then } s \text{ else } s \sim \text{ if } b \text{ then } t \text{ else } t} \quad \text{CS}
\]

Lemma

From a proof of \( a, s \sim b, t \) we can extract a smaller proof of \( s \sim t \).
Proof Cut: Introduction of a Conditional on Both Sides

\[
\begin{align*}
& a, s \sim b, t \\
\text{if } a \text{ then } & s \text{ else } s \sim \text{ if } b \text{ then } t \text{ else } t \\
\text{CS} & \\
R \\
& s \sim t
\end{align*}
\]

Lemma

From a proof of \( a, s \sim b, t \) we can extract a smaller proof of \( s \sim t \).

\( \Rightarrow \) Proof Cut Elimination
Proof Cut

\[
\frac{a_1, b_2, b_3, u_4, w_5, u_6, v_7 \sim d_1, c_2, d_3, s_4, t_5, r_6, p_7}{\text{FA}^{(3)}}
\]

where \( p \equiv \text{if } c \text{ then } s \text{ else } t \)

\[\begin{align*}
\text{if } a \text{ then } u \text{ else } v & \sim \text{if } c \text{ then } s \text{ else } t \\
\end{align*}\]
Decision Procedure

Proof Cut

\[
\frac{a_1, b_2, b_3, u_4, w_5, u_6, v_7 \sim d_1, c_2, d_3, s_4, t_5, r_6, p_7}{\text{FA}^{(3)}}
\]

\[
\begin{array}{c}
\frac{a_1}{b_2} \\
\frac{v_7}{u_4} \\
\frac{b_3}{w_5} \\
\frac{u_6}{-d_1} \\
\frac{c_2}{s_4} \\
\frac{d_3}{t_5} \\
\frac{p_7}{r_6} \\
\end{array}
\]

if \( a \) then \( u \) else \( v \) \( \sim \) if \( c \) then \( s \) else \( t \)

where \( p \equiv \text{if } c \text{ then } s \text{ else } t \)

Key Lemma

If \( b, b' \sim b'', b'' \) can be shown using only FA, Dup and CCA1 then \( b' \equiv b'' \).
**Decision Procedure**

**Proof Cut**

\[
\begin{align*}
\frac{a_1, b_2, b_3, u_4, w_5, u_6, v_7 \sim d_1, c_2, d_3, s_4, t_5, r_6, p_7}{ \text{FA}^{(3)}}
\end{align*}
\]

\[
\begin{array}{c}
\frac{a_1}{b_2} \\
\frac{u_4}{w_5}
\end{array}
\frac{v_7}{b_3}
\frac{u_6}{} \sim
\frac{c_2}{s_4} \\
\frac{d_1}{d_3} \\
\frac{t_5}{r_6} \\
\frac{p_7}{}
\]

\[
\frac{\text{if } a \text{ then } u \text{ else } v \sim \text{if } c \text{ then } s \text{ else } t}{R}
\]

where \( p \equiv \text{if } c \text{ then } s \text{ else } t \)

**Proof Cut Elimination**

\[
\bullet \ b_2, b_3 \sim c_2, d_3 \quad \Rightarrow \quad c \equiv d.
\]
Proof Cut

\[
\frac{a_1, b_2, b_3, u_4, w_5, u_6, v_7 \sim d_1, c_2, d_3, s_4, t_5, r_6, p_7}{\text{FA}^{(3)}}
\]

\[
\begin{array}{c}
\text{if } a \text{ then } u \text{ else } v \\
\text{if } c \text{ then } s \text{ else } t
\end{array}
\]

where \( p \equiv \text{if } c \text{ then } s \text{ else } t \)

Proof Cut Elimination

- \( b_2, b_3 \sim c_2, d_3 \quad \Rightarrow \quad c \equiv d \)
- \( a_1, b_2 \sim d_1, c_2 \quad \Rightarrow \quad a \equiv b \)
Theorem

The following problem is decidable:

**Input:** A game $\vec{u} \sim \vec{v}$.

**Question:** Is there a sequence of game transformations in $Ax$ showing that $\vec{u} \sim \vec{v}$ is secure?
Theorem

The following problem is decidable:

**Input:** A game $\vec{u} \sim \vec{v}$.

**Question:** Is there a sequence of game transformations in $\text{Ax}$ showing that $\vec{u} \sim \vec{v}$ is secure?

Remark: Basic Games

The above result holds when using CCA2 as basic games.
Strategy: Theorem

Theorem

The following problem is decidable:
Input: A game $\vec{u} \sim \vec{v}$.
Question: Is there a sequence of game transformations in Ax showing that $\vec{u} \sim \vec{v}$ is secure?

Remark: Basic Games

The above result holds when using CCA2 as basic games.

Sketch

- Commute rule applications to order them as follows:

$$ (2Box + R_{\square}) \cdot CS_{\square} \cdot FA_{if} \cdot FA_{f} \cdot Dup \cdot U $$

- We do proof cut eliminations to get a small proof.
Introduction

The Model

Game Transformations
- Basic Games
- Game Transformations

Decision Result

Conclusion
## Conclusion

### Our Works
- Designed and proved correct a set of game transformations.
- Showed a decision result for this set of game transformations.

### Advantages and Drawbacks
- **Full automation.**
- **Completeness:** absence of proof implies the existence of an attack.
- **Bounded number of sessions.**
- **Cannot easily add cryptographic assumptions:** current result only of CCA2.

### Future Works
- Support for a large class of primitives and associated assumptions.
- Interactive/automatic prover using the strategy.
Conclusion

**Our Works**
- Designed and proved correct a set of game transformations.
- Showed a decision result for this set of game transformations.

**Advantages and Drawbacks**
- Full automation.
- Completeness: absence of proof implies the existence of an attack.
- Bounded number of sessions.
- Cannot easily add cryptographic assumptions: current result only of CCA2.
Conclusion

Our Works

- Designed and proved correct a set of game transformations.
- Showed a decision result for this set of game transformations.

Advantages and Drawbacks

- Full automation.
- Completeness: absence of proof implies the existence of an attack.
- Bounded number of sessions.
- Cannot easily add cryptographic assumptions: current result only of CCA2.

Future Works

- Support for a large class of primitives and associated assumptions.
- Interactive/automatic prover using the strategy.
Thanks for your attention