Automated Proof for Formal Indistinguishability and Its Applications (Work in Progress)

Long Ngo, Colin Boyd and Juan Gonzalez Nieto

{l.t.ngo, c.boyd, j.gonzaleznielo}@qut.edu.au

Queensland University of Technology

June 30, 2011
Outline

Motivation

How it can be done

Applications
Motivation

Are these two sequences of information computationally indistinguishable?

\[ \{ g, g^{X_1}, g^{X_2}, g^{X_3}, g^{X_1X_2}, g^{X_2X_3}, g^{X_1X_3}, \{ K_1 \} \}_{h(g^{X_1X_2X_3})}, \{ K_2 \}_{K_1}, \{ m \}_{K_2} \]

and

\[ \{ g, g^{X_1}, g^{X_2}, g^{X_3}, g^{X_1X_2}, g^{X_2X_3}, g^{X_1X_3}, \{ K_1 \} \}_{h(g^{X_1X_2X_3})}, \{ K_2 \}_{K_1}, \{ 0 \}_{K_2} \]

We want a tool that can quickly answer such a question.

The tool should be easily extended.

Applications:

- Weak security: Eavesdropper, IND-CPA attacker
- Strong security: can be achieved by generic transformation from weak to strong security.
Outline

Motivation

How it can be done

Applications
Model: Syntax

- A **signature** \( \Sigma = (S, \leq_S, \mathcal{F}) \) consists of a countably infinite set of **sorts** \( S \) with partial order \( \leq_S \), and a finite set of **functions** \( \mathcal{F} \).
- There are three countably infinite sets \( N, \mathcal{X} \) and \( P \), where \( N \) is a set of names, \( \mathcal{X} \) is a set of first-order variables and \( P \) is a set of second-order variables.
- A term is
  \[
  T ::= x \\
  \mid n \\
  \mid p(T_1, \ldots, T_k) \\
  \mid f(T_1, \ldots, T_k)
  \]
- A frame is a sequence of terms
- Behaviours of functions are model by equational theories.
- We use ordered-sorts in order to capture the fact that \( a \) and \( a \parallel b \) can be different sorts but \( h(a) \) and \( h(a \parallel b) \) are both valid (\( h \) is a hash).
Model: Semantics

The language has a computational semantics

- Each sort is associated with a set of bitstrings
- Each function symbol is associated with a real function
- Each name is evaluated to a nonce of its sort.
- Each second-order variable is evaluated by the adversary
- Then terms and frames are evaluated recursively.
Computational Soundness: Static Equivalence vs Formal Indistinguishability (FIR) \(^1\)

- Static Equivalence means the symbolic attacker cannot distinguish between two frames by using the equivalence theories (which abstract the behaviours of functions).
- Static Equivalence: start from what the attacker can do (modeled symbolically) to show what the attacker cannot do.
- Formal Indistinguishability: start from what the attacker cannot do to show what else the attacker cannot do.
- FIR is closer to computational indistinguishability, which is based on a set of hard problem assumption.
- Example: \(\{x_1 = g, x_2 = g^a, x_3 = g^b, x_4 = g^{a^2*b^2}\}\) and \(\{x_1 = g, x_2 = g^a, x_3 = g^b, x_4 = g^c\}\) are statically equivalent, but not necessarily implied by the DDH assumption.

\(^1\) proposed by Bana

Long Ngo, Colin Boyd and Juan Gonzalez Nieto

Automated Proof for Formal Indistinguishability and Its Applications (Work in Progress)
Formal Indistinguishability Definitions

Given an equational theory $\equiv_E$, FIR is defined as:

1. $(GE1)$ If $\phi_1 \equiv \phi_2$ then $\phi \phi_1 \equiv \phi \phi_2$ for any frame $\phi$ such that $\text{var}(\phi) \subseteq \text{dom}(\phi_i)$ and $\text{name}(\phi) \cap \text{name}(\phi_i) = \emptyset$.
2. $(GE2)$ $\phi_1 \equiv \phi_2$ if $\phi_1 =_E \phi_2$.
3. $(GE3)$ $\tau(\phi) \equiv \phi$ for any renaming $\tau$.

The first rule captures the fact: for any algorithm $A$, if $x \equiv y$, then $A(x) \equiv A(y)$.

If the generating set $S_i$ is computationally sound, then FIR is computationally sound.
Example

One may start from \( S_i = \{ \{ g, g^a, g^b, g^{ab} \} \cong \{ g, g^a, g^b, g^c \} \} \) to generate \( \{ \{ g, g^a, g^b, 2g^{ab} \} \cong \{ g, g^a, g^b, 2g^c \} \} \)

Can this be generated?

\[
\{ g, g^{X_1}, g^{X_2}, g^{X_3}, g^{X_1X_2}, g^{X_2X_3}, g^{X_1X_3}, \{ K_1 \}_h(g^{X_1X_2X_3}), \{ K_2 \} K_1, \{ m \} K_2 \} \\
\cong \\
\{ g, g^{X_1}, g^{X_2}, g^{X_3}, g^{X_1X_2}, g^{X_2X_3}, g^{X_1X_3}, \{ K_1 \} h(g^{X_1X_2X_3}), \{ K_2 \} K_1, \{ 0 \} K_2 \}
\]
Formal Indistinguishability with Hash

- Informally, $h(x) \cong r$ if $x$ is non-derivable (Random Oracle Model).
- Because of hash, the concept of **Formal Non-Derivability** (FNDR) is proposed by Ene et al.
- Similarly to FIR, there are 7 closure rules to generate a set of FNDR from an initial set $S_d$, which models a set of hardness assumption.
- Example: One may start from $S_d = \{\{g, g^a, g^b\} \not\sim g^c\}$ to generate $\{\{g, g^a, g^b\} \cong 2g^c\}$
How can we automatically check FIR

- FIR is an transitive, so we can use an abstract rewriting system for reasoning.
- Term rewriting cannot be used because FIR is a relation on frames and not a congruence.
- We define a rewriting system called FIR-frame rewriting system, which is tailored to suits FIR
- Then the check can be automated, provided that the rewriting system is convergent.
- However, if a frame contains hash, then we have to check non-derivability.
How can we automatically check FNDR

- Automatic check for FNDR is more tricky, because FNDR is not transitive.
- Example: \( \{g, g^a, g^b\} \not\sim g^c \) and \( \{g^c\} \not\sim g^a \), but \( \{g, g^a, g^b\} \not\sim g^a \) is wrong.
- Solution: Design a semi-decision procedure, which covers most of practical cases.
- **Work is still in progress.**
Outline

Motivation

How it can be done

Applications
IND-CPA Encryption Schemes

- IND-CPA Security of an encryption scheme can be represented as the indistinguishability between two frames, which have second-order variables.

- Example: The security of ElGamal scheme is expressed as:
  \[
  \{ x_0 = g, x_1 = g^a, x_2 = g^b, x_3 = g^{a*b} \cdot p(g, g^a) \} \sim \{ x_0 = g, x_1 = g^a, x_2 = s, x_3 = t \}
  \]

- If you want IND-CCA scheme, there exists a method to transform an IND-CPA to IND-CCA encryption scheme.
Key exchange protocol

- Passive security of key exchange protocol can also be expressed as the indistinguishability between two frames

- Example: Security of a tripartite DH key exchange protocol can be expressed as

\[ \{g, g^{X_1}, g^{X_2}, g^{X_3}, g^{X_1X_2}, g^{X_2X_3}, g^{X_1X_3}, g^{X_1X_2X_3}\} \sim \{g, g^{X_1}, g^{X_2}, g^{X_3}, g^{X_1X_2}, g^{X_2X_3}, g^{X_1X_3}, g^c\} \]

- Also, there exist a method which transform a pasively secure key exchange to an actively secure one.
Scalability

- Unlike static equivalence, combine signatures in FIR is simple.
- As long as the FIR in each signature is computationally sound, then the FIR in the combined signature is also computationally sound.
- This is from the fact that the computational soundness depends only on the generating sets.
Conclusion

- A potential tool for automated proofs of computational indistinguishability is presented.
- We use the notion Formal Indistinguishability, which is extended to cover ordered sorts.
- The tool has great scalability due to the easiness to combine signatures.
- Potential application of the tool is presented.
- The work is still in progress. Comments are extremely welcome.