Standard secure encryption under stronger forms of attacks, with applications to computational soundness

Mohammad Hajiabadi, Bruce Kapron
Computer Science Department
University of Victoria

December 3, 2013

What I am going to present

Encryption security and stronger attack models

- KDM attack models
- Adaptive corruption attacks
- What can we show?

Computational soundness of symbolic security

Overview of standard semantic security

- Syntax of Public-key encryption: $\mathcal{E} = (Gen, Enc, Dec)$

Overview of standard semantic security

- Syntax of Public-key encryption: $\mathcal{E} = (Gen, Enc, Dec)$
  - Key generation: $(pk, sk) \leftarrow K(1^n)$;
  - Encryption: $c \leftarrow E_{pk}(m)$;
  - Decryption: $D_{sk}(c) = m$.

Overview of standard semantic security

- Syntax of Public-key encryption: $\mathcal{E} = (Gen, Enc, Dec)$
  - Key generation: $(pk, sk) \leftarrow K(1^n)$;
  - Encryption: $c \leftarrow E_{pk}(m)$;
  - Decryption: $D_{sk}(c) = m$.
- Semantic (CPA) security: For every PPT $\mathcal{A}$:
  - $(pk, sk) \leftarrow G(1^n)$
  - $(m_0, m_1) \leftarrow \mathcal{A}(pk)$;
  - $|\Pr[\mathcal{A}(Enc_{pk}(m_0), pk) = 1] - \Pr[\mathcal{A}(Enc_{pk}(m_1), pk) = 1]| = negl$

Circular security

- l-circular security: $(E_{pk_1}(sk_2), \ldots, E_{pk_l}(sk_1))$ looks as good as $(E_{pk_1}(r_1), \ldots, E_{pk_l}(r_l))$.

Circular security

- l-circular security: $(E_{pk_1}(sk_2), \ldots, E_{pk_l}(sk_1))$ looks as good as $(E_{pk_1}(r_1), \ldots, E_{pk_l}(r_l))$.
- What is known:
  - for any l, semantic security $\not\Rightarrow$ l-circular security (using obfuscation techniques) [Koppula-Ramchen-Waters eprint-2013]

Circular security

- l-circular security: $(E_{pk_1}(sk_2), \ldots, E_{pk_l}(sk_1))$ looks as good as $(E_{pk_1}(r_1), \ldots, E_{pk_l}(r_l))$.
- What is known:
  - for any l, semantic security $\not\Rightarrow$ l-circular security (using obfuscation techniques) [Koppula-Ramchen-Waters eprint-2013]
  - $(E_{pk_1}(sk_2), \ldots, E_{pk_l}(sk_1))$ reveals all sk_i's! [Koppula-Ramchen-Waters eprint-2013]

What we want to do today

- $(pk_1, sk_1), \ldots, (pk_l, sk_l)$

What we want to do today

- $(pk_1, sk_1), \ldots, (pk_l, sk_l)$
- Sequence of KDM queries ($E_{pk_i}(sk_j)$ or $E_{pk_i}(E_{pk_j}(sk_r))$, etc.)

What we want to do today

- $(pk_1, sk_1), \ldots, (pk_l, sk_l)$
- Sequence of KDM queries ($E_{pk_i}(sk_j)$ or $E_{pk_i}(E_{pk_j}(sk_r))$, etc.)
- Interleaving corruption queries (corrupt(sk_i))

What we want to do today

- $(pk_1, sk_1), \ldots, (pk_l, sk_l)$
- Sequence of KDM queries ($E_{pk_i}(sk_j)$ or $E_{pk_i}(E_{pk_j}(sk_r))$, etc.)
- Interleaving corruption queries ($\text{corrupt}(sk_i)$)
- Goal: Proving secrecy of non-corrupted keys under the CPA assumption.

What we want to do today

- $(pk_1, sk_1), \ldots, (pk_l, sk_l)$
- Sequence of KDM queries ($E_{pk_i}(sk_j)$ or $E_{pk_i}(E_{pk_j}(sk_r))$, etc.)
- Interleaving corruption queries (corrupt(sk_i))
- Goal: Proving secrecy of non-corrupted keys under the CPA assumption.

How we are going to do it:

- Sequence of games, where each game:

What we want to do today

- $(pk_1, sk_1), \ldots, (pk_l, sk_l)$
- Sequence of KDM queries ($E_{pk_i}(sk_j)$ or $E_{pk_i}(E_{pk_j}(sk_r))$, etc.)
- Interleaving corruption queries (corrupt(sk_i))
- Goal: Proving secrecy of non-corrupted keys under the CPA assumption.

How we are going to do it:

- Sequence of games, where each game:
  - multiple key based: $(pk_1, sk_1) \ldots, (pk_l, sk_l)$

What we want to do today

- $(pk_1, sk_1), \ldots, (pk_l, sk_l)$
- Sequence of KDM queries ($E_{pk_i}(sk_j)$ or $E_{pk_i}(E_{pk_j}(sk_r))$, etc.)
- Interleaving corruption queries (corrupt(sk_i))
- Goal: Proving secrecy of non-corrupted keys under the CPA assumption.

How we are going to do it:

- Sequence of games, where each game:
  - multiple key based: $(pk_1, sk_1) \ldots, (pk_l, sk_l)$
  - Consists of two phases:

What we want to do today

- $(pk_1, sk_1), \ldots, (pk_l, sk_l)$
- Sequence of KDM queries ($E_{pk_i}(sk_j)$ or $E_{pk_i}(E_{pk_j}(sk_r))$, etc.)
- Interleaving corruption queries (corrupt(sk_i))
- Goal: Proving secrecy of non-corrupted keys under the CPA assumption.

How we are going to do it:

- Sequence of games, where each game:
  - multiple key based: $(pk_1, sk_1) \ldots, (pk_l, sk_l)$
  - Consists of two phases:
    - First phase: $\mathcal{A}$ gets to obtain some info about sk_i's through KDM and corruption queries

What we want to do today

- $(pk_1, sk_1), \ldots, (pk_l, sk_l)$
- Sequence of KDM queries ($E_{pk_i}(sk_j)$ or $E_{pk_i}(E_{pk_j}(sk_r))$, etc.)
- Interleaving corruption queries (corrupt(sk_i))
- Goal: Proving secrecy of non-corrupted keys under the CPA assumption.

How we are going to do it:

- Sequence of games, where each game:
  - multiple key based: $(pk_1, sk_1) \ldots, (pk_l, sk_l)$
  - Consists of two phases:
    - First phase: $\mathcal{A}$ gets to obtain some info about sk_i's through KDM and corruption queries
    - Second phase: $\mathcal{A}$ participates in a standard indist experiment.

Some notation

- We denote $Enc_{pk_i}(sk_j)$ as $\{sk_j\}_{pk_i}$.

Some notation

- We denote $Enc_{pk_i}(sk_j)$ as $\{sk_j\}_{pk_i}$.
- Nested encryptions: $Enc_{pk_1}(Enc_{pk_2}(sk_3))$ as $\{\{sk_3\}_{pk_2}\}_{pk_1}$.

Some notation

- We denote $Enc_{pk_i}(sk_j)$ as $\{sk_j\}_{pk_i}$.
- Nested encryptions: $Enc_{pk_1}(Enc_{pk_2}(sk_3))$ as $\{\{sk_3\}_{pk_2}\}_{pk_1}$.
- What do I mean by a key cycle:

Some notation

- We denote $Enc_{pk_i}(sk_j)$ as $\{sk_j\}_{pk_i}$.
- Nested encryptions: $Enc_{pk_1}(Enc_{pk_2}(sk_3))$ as $\{\{sk_3\}_{pk_2}\}_{pk_1}$.
- What do I mean by a key cycle:
  - $(\{sk_1\}_{pk_2}, \{sk_2\}_{pk_1})$ is a key cycle;
  - $\{\{sk_1\}_{pk_2}\}_{pk_1}$ is also a key cycle!

Some notation

- We denote $Enc_{pk_i}(sk_j)$ as $\{sk_j\}_{pk_i}$.
- Nested encryptions: $Enc_{pk_1}(Enc_{pk_2}(sk_3))$ as $\{\{sk_3\}_{pk_2}\}_{pk_1}$.
- What do I mean by a key cycle:
  - $(\{sk_1\}_{pk_2}, \{sk_2\}_{pk_1})$ is a key cycle;
  - $\{\{sk_1\}_{pk_2}\}_{pk_1}$ is also a key cycle!
- No key cycle = ordering $(sk_1, \ldots, sk_l)$ s.t. every plaintext occurrence of sk_i is encrypted under $\{pk_1, \ldots, pk_{i-1}\}$.

Game1 : A priori known encryption ordering

- First phase:
  - A priori known fixed ordering $\langle sk_1, \ldots, sk_n \rangle$:

Game1 : A priori known encryption ordering

- First phase:
  - A priori known fixed ordering $\langle sk_1, \ldots, sk_n \rangle$:
    - $\mathcal{A}$ may obtain encryptions of any sk_i under any key in $\{pk_1, \ldots, pk_{i-1}\}$

Game1 : A priori known encryption ordering

- First phase:
  - A priori known fixed ordering $\langle sk_1, \ldots, sk_n \rangle$:
    - $\mathcal{A}$ may obtain encryptions of any sk_i under any key in $\{pk_1, \ldots, pk_{i-1}\}$
  - No corruptions.
- Second phase: Choose any pk_j; LOR interaction.

Game1 : A priori known encryption ordering

- First phase:
  - A priori known fixed ordering $\langle sk_1, \ldots, sk_n \rangle$:
    - $\mathcal{A}$ may obtain encryptions of any sk_i under any key in $\{pk_1, \ldots, pk_{i-1}\}$
  - No corruptions.
- Second phase: Choose any pk_j; LOR interaction.

A simple hybrid argument: Game1-security = semantic security.

Game1 : A priori known encryption ordering

- First phase:
  - A priori known fixed ordering $\langle sk_1, \ldots, sk_n \rangle$:
    - $\mathcal{A}$ may obtain encryptions of any sk_i under any key in $\{pk_1, \ldots, pk_{i-1}\}$
  - No corruptions.
- Second phase: Choose any pk_j; LOR interaction.

A simple hybrid argument: Game1-security = semantic security.

Game2 = Game1+ the encryption ordering is adaptively made by $\mathcal{A}$ (i.e., *a priori* unknown).

Game1 : A priori known encryption ordering

- First phase:
  - A priori known fixed ordering $\langle sk_1, \ldots, sk_n \rangle$:
    - $\mathcal{A}$ may obtain encryptions of any sk_i under any key in $\{pk_1, \ldots, pk_{i-1}\}$
  - No corruptions.
- Second phase: Choose any pk_j; LOR interaction.

A simple hybrid argument: Game1-security = semantic security.

Game2 = Game1+ the encryption ordering is adaptively made by $\mathcal{A}$ (i.e., *a priori* unknown).
Is security under Game2 = semantic security?

Game1 : A priori known encryption ordering

- First phase:
  - A priori known fixed ordering $\langle sk_1, \ldots, sk_n \rangle$:
    - $\mathcal{A}$ may obtain encryptions of any sk_i under any key in $\{pk_1, \ldots, pk_{i-1}\}$
  - No corruptions.
- Second phase: Choose any pk_j; LOR interaction.

A simple hybrid argument: Game1-security = semantic security.

Game2 = Game1+ the encryption ordering is adaptively made by $\mathcal{A}$ (i.e., *a priori* unknown).
Is security under Game2 = semantic security?

- We don't know. (discuss partial results later)

benign circular encryption

Question: Benign forms of key cycles?

benign circular encryption

Question: Benign forms of key cycles?

Example 1: $\{sk_1\}_{pk_2}, \{sk_2\}_{pk_1}$ is *not* benign.

Example 2: $\{\{sk_1\}_{pk_2}\}_{pk_1}$ is benign.

benign circular encryption

Question: Benign forms of key cycles?

Example 1: $\{sk_1\}_{pk_2}, \{sk_2\}_{pk_1}$ is *not* benign.

Example 2: $\{\{sk_1\}_{pk_2}\}_{pk_1}$ is benign.

Question: So what is the structure?

New interpretation of ordering

- Fix ordering $\langle sk_1, \ldots, sk_n \rangle$.

New interpretation of ordering

- Fix ordering $\langle sk_1, \ldots, sk_n \rangle$.
- Rule: if sk_i is every encrypted, at least *one* of the encryption keys is in $\{pk_1, \ldots, pk_{i-1}\}$.

New interpretation of ordering

- Fix ordering $\langle sk_1, \ldots, sk_n \rangle$.
- Rule: if sk_i is every encrypted, at least *one* of the encryption keys is in $\{pk_1, \ldots, pk_{i-1}\}$.

✓ in $\{\{sk_1\}_{pk_2}\}_{pk_1}$ respects this rule; (ie $\langle sk_2, sk_1 \rangle$)

× In $\{sk_1\}_{pk_2}, \{sk_2\}_{pk_1}$ doesn't.

Benign cyclic encryption

Game3: fixed ordering $\langle sk_1, \ldots, sk_n \rangle$.

Benign cyclic encryption

Game3: fixed ordering $\langle sk_1, \ldots, sk_n \rangle$.

- : First phase: key-dependent encryptions that respects the ordering
  - ✓ $\{\{sk_i\}_{pk_i}\}_{pk_{i-1}}$
  - × $\{sk_i\}_{pk_i}$
- No corruption.
- Second phase: like before.

Then

Benign cyclic encryption

Game3: fixed ordering $\langle sk_1, \ldots, sk_n \rangle$.

- : First phase: key-dependent encryptions that respects the ordering
  - ✓ $\{\{sk_i\}_{pk_i}\}_{pk_{i-1}}$
  - × $\{sk_i\}_{pk_i}$
- No corruption.
- Second phase: like before.

Then

Security under Game3 = semantic security.

$(pk_1, sk_1), \ldots, (pk_n, sk_n)$.

Goal: No restriction in the first phase!

- Definition: Call $S \subseteq \{sk_1, \ldots, sk_n\}$ *safe* if S admits an ordering respected by adversary's queries.

$(pk_1, sk_1), \ldots, (pk_n, sk_n)$.

Goal: No restriction in the first phase!

- Definition: Call $S \subseteq \{sk_1, \ldots, sk_n\}$ *safe* if S admits an ordering respected by adversary's queries.
  - $\langle sk_{i_1}, \ldots, sk_{i_p} \rangle$ s.t. sk_{i_r} is always encrypted under one of $\{pk_{i_1}, \ldots, pk_{i_{r-1}}\}$, where $S = \{sk_{i_1}, \ldots, sk_{i_p}\}$.

$(pk_1, sk_1), \ldots, (pk_n, sk_n)$.

Goal: No restriction in the first phase!

- Definition: Call $S \subseteq \{sk_1, \ldots, sk_n\}$ *safe* if S admits an ordering respected by adversary's queries.
  - $\langle sk_{i_1}, \ldots, sk_{i_p} \rangle$ s.t. sk_{i_r} is always encrypted under one of $\{pk_{i_1}, \ldots, pk_{i_{r-1}}\}$, where $S = \{sk_{i_1}, \ldots, sk_{i_p}\}$.

Fact: The set of all safe S's admits a *greatest* set.

$(pk_1, sk_1), \ldots, (pk_n, sk_n)$.

Goal: No restriction in the first phase!

- Definition: Call $S \subseteq \{sk_1, \ldots, sk_n\}$ *safe* if S admits an ordering respected by adversary's queries.
  - $\langle sk_{i_1}, \ldots, sk_{i_p} \rangle$ s.t. sk_{i_r} is always encrypted under one of $\{pk_{i_1}, \ldots, pk_{i_{r-1}}\}$, where $S = \{sk_{i_1}, \ldots, sk_{i_p}\}$.

Fact: The set of all safe S's admits a *greatest* set.

This maximal safe set (call MS) is the set of keys we want to show they remain "secure".

$(pk_1, sk_1), \ldots, (pk_n, sk_n)$.

Goal: No restriction in the first phase!

- Definition: Call $S \subseteq \{sk_1, \ldots, sk_n\}$ *safe* if S admits an ordering respected by adversary's queries.
  - $\langle sk_{i_1}, \ldots, sk_{i_p} \rangle$ s.t. sk_{i_r} is always encrypted under one of $\{pk_{i_1}, \ldots, pk_{i_{r-1}}\}$, where $S = \{sk_{i_1}, \ldots, sk_{i_p}\}$.

Fact: The set of all safe S's admits a *greatest* set.

This maximal safe set (call MS) is the set of keys we want to show they remain "secure".

Example:

- First phase: $\{sk_1\}_{pk_2}$, $\{sk_2\}_{pk_1}$, $\{\{\{sk_3\}_{pk_3}\}_{pk_2}\}_{pk_4}$, $\{sk_4\}_{pk_5}$
- Second phase: $\{sk_4, sk_5\}$ is the maximal safe set.

$(pk_1, sk_1), \ldots, (pk_n, sk_n)$.

Goal: No restriction in the first phase!

- Definition: Call $S \subseteq \{sk_1, \ldots, sk_n\}$ *safe* if S admits an ordering respected by adversary's queries.
  - $\langle sk_{i_1}, \ldots, sk_{i_p} \rangle$ s.t. sk_{i_r} is always encrypted under one of $\{pk_{i_1}, \ldots, pk_{i_{r-1}}\}$, where $S = \{sk_{i_1}, \ldots, sk_{i_p}\}$.

Fact: The set of all safe S's admits a *greatest* set.

This maximal safe set (call MS) is the set of keys we want to show they remain "secure".

Example:

- First phase: $\{sk_1\}_{pk_2}$, $\{sk_2\}_{pk_1}$, $\{\{\{sk_3\}_{pk_3}\}_{pk_2}\}_{pk_4}$, $\{sk_4\}_{pk_5}$
- Second phase: $\{sk_4, sk_5\}$ is the maximal safe set.

The remaining keys have occurred in key cycles like:

- $\{sk_1\}_{pk_2}, \ldots, \{sk_i\}_{pk_1}$
- $\{\{sk_1\}_{pk_1}\}_{pk_1}$
- $\{\{sk_1\}_{pk_2}\}_{pk_2}, \{sk_2\}_{pk_1}$

- Final strengthening: Adaptive corruption in the first phase.

- Final strengthening: Adaptive corruption in the first phase.
- The notion of a safe set extends easily.

Final game

Again over keys $(pk_1, sk_1) \ldots, (pk_n, sk_n)$, and in two phases:

Final game

Again over keys $(pk_1, sk_1) \ldots, (pk_n, sk_n)$, and in two phases:

- First phase: Key-dependent encryptions+adaptive corruptions (No restrictions)

Final game

Again over keys $(pk_1, sk_1) \ldots, (pk_n, sk_n)$, and in two phases:

- First phase: Key-dependent encryptions+adaptive corruptions (No restrictions)
- Second phase: LOR indist for the maximal safe set.

Final game

Again over keys $(pk_1, sk_1) \ldots, (pk_n, sk_n)$, and in two phases:

- First phase: Key-dependent encryptions+adaptive corruptions (No restrictions)
- Second phase: LOR indist for the maximal safe set.

We call this notion RC-security (restricted circular security).

Our results

- Question: Is RC-security implied by CPA security?

Our results

- Question: Is RC-security implied by CPA security?
- Previous results: Panjwani (TCC 2007) shows a reduction $O(n^l)$ for: single encryptions+absence of key cycles.

Our results

- Question: Is RC-security implied by CPA security?
- Previous results: Panjwani (TCC 2007) shows a reduction $O(n^l)$ for: single encryptions+absence of key cycles.
  - l: length of the longest encryption path.

Our results

- Question: Is RC-security implied by CPA security?
- Previous results: Panjwani (TCC 2007) shows a reduction $O(n^l)$ for: single encryptions+absence of key cycles.
  - l: length of the longest encryption path.
- By building on Panjwani's work, we show if the diameter of the induced subgraph on the "maximal safe set" is constant, RC security is implied by CPA security.

Our results

- Question: Is RC-security implied by CPA security?
- Previous results: Panjwani (TCC 2007) shows a reduction $O(n^l)$ for: single encryptions+absence of key cycles.
  - l: length of the longest encryption path.
- By building on Panjwani's work, we show if the diameter of the induced subgraph on the "maximal safe set" is constant, RC security is implied by CPA security.
- We next generalize it to the CCA2 setting for applications to computationally soundsymbolic security (described next).

Our results

- Question: Is RC-security implied by CPA security?
- Previous results: Panjwani (TCC 2007) shows a reduction $O(n^l)$ for: single encryptions+absence of key cycles.
  - l: length of the longest encryption path.
- By building on Panjwani's work, we show if the diameter of the induced subgraph on the "maximal safe set" is constant, RC security is implied by CPA security.
- We next generalize it to the CCA2 setting for applications to computationally soundsymbolic security (described next).

Extensions and Open Questions

- Improving the $O(n^l)$-reduction factor.

Extensions and Open Questions

- Improving the $O(n^l)$-reduction factor.
- Enhancing KDM security with adaptive corruptions.

Extensions and Open Questions

- Improving the $O(n^l)$-reduction factor.
- Enhancing KDM security with adaptive corruptions.
  - This would enable secure realizations of protocols with inductive (as opposed to coinductive), symbolic security proofs.

Overview

1. Computational cryptography
   - Cryptographic primitives are modeled as PPT algorithms,
   - Security holds against poly-time adversaries.
2. Symbolic security (Dolev-Yao models)
   - High-level abstractions of cryptographic primitives,
   - (non-deterministic) symbolic adversaries: following certain symbolic rules.
   - Much easier proofs (due to abstractions), Allowing automation,

Relating the two views

Goal: Achieving the best of the two worlds.

One possible approach:

- *Computational Soundness:* Allowing to obtain computational security guarantees from symbolic proofs.

Relating the two views

Goal: Achieving the best of the two worlds.

One possible approach:

- *Computational Soundness:* Allowing to obtain computational security guarantees from symbolic proofs.
- Typical form: If protocol Π is symbolically secure $\Rightarrow$ generic instantiations of Π (under exactly-defined secure primitives) are computationally secure.

Relating the two views

Goal: Achieving the best of the two worlds.

One possible approach:

- *Computational Soundness:* Allowing to obtain computational security guarantees from symbolic proofs.
- Typical form: If protocol Π is symbolically secure $\Rightarrow$ generic instantiations of Π (under exactly-defined secure primitives) are computationally secure.

This enables:

- Doing proofs in a symbolic model (without explicitly dealing with complexity-based notions), and
- obtaining computational security from (once and for all) established computational soundness theorems.

What we demand

We want from soundness:

- Not too demanding assumptions (e.g, not rely on random-oracles, etc.),
- Applicable to large classes of protocols and security properties,
- ...

Prior work

- Abadi & Rogaway 2001: Pioneering work. Limited to eavesdropping adversaries and single-message protocols. Many extensions since then in the eavesdropping setting ([AJ'2001], [MW'2002], [H'2004], . . .)

Prior work

- Abadi & Rogaway 2001: Pioneering work. Limited to eavesdropping adversaries and single-message protocols. Many extensions since then in the eavesdropping setting ([AJ'2001], [MW'2002], [H'2004], . . .)
- Micciancio, Warinschi TCC 2004:
  - Active adversaries,
  - Discussing general types of security: trace-based security properties (e.g., entity authentication [BR-Crypto 94])

  Assumptions in Micciancio & Warinschi framework:
  - static corruption (all corruptions are made nonadaptively at the beginning),
  - secret keys cannot be part of messages.

Prior work

- Abadi & Rogaway 2001: Pioneering work. Limited to eavesdropping adversaries and single-message protocols. Many extensions since then in the eavesdropping setting ([AJ'2001], [MW'2002], [H'2004], . . .)
- Micciancio, Warinschi TCC 2004:
  - Active adversaries,
  - Discussing general types of security: trace-based security properties (e.g., entity authentication [BR-Crypto 94])

  Assumptions in Micciancio & Warinschi framework:
  - static corruption (all corruptions are made nonadaptively at the beginning),
  - secret keys cannot be part of messages.

Our work: Trying to relax both assumptions above.

Some assumptions (Informal)

Assumptions used in our soundness theorem:

Some assumptions (Informal)

Assumptions used in our soundness theorem:

- Assumptions on protocols:
  - symmetric and asymmetric encryption as the only primitives.
  - protocols admit a symbolic specification. (e.g., NSL protocol: $(\{A, N_A\}_{k_B}, \{N_A, N_B, B\}_{k_A}, \{N_B\}_{k_B})$).
  - We allow secret keys to be part of messages.

Some assumptions (Informal)

Assumptions used in our soundness theorem:

- Assumptions on protocols:
  - symmetric and asymmetric encryption as the only primitives.
  - protocols admit a symbolic specification. (e.g., NSL protocol: $(\{A, N_A\}_{k_B}, \{N_A, N_B, B\}_{k_A}, \{N_B\}_{k_B})$).
  - We allow secret keys to be part of messages.
- Adversarial assumptions:
  - Active adversary with adaptively corrupting power.

Active adversaries and secret keys being part of messages

Question: What happens if we allow secret keys to be part of messages?

1. It may lead to the creation of key cycles.
2. It may lead to the creation of some form of (a priori unknown) encryption-ordering between keys.

We explain further about these points through an example.

Motivating example

Consider the following protocol over A, B, C with public keys k_A, k_B, k_C:

$$A \to B: \ (\{k_1\}_{k_B}, \{k_2\}_{k_B})$$
$$B \to C: \ (\{k_1\}_{k_C}, \{k_2\}_{k_1})$$

k_1, k_2: Local session keys.

Motivating example

Consider the following protocol over A, B, C with public keys k_A, k_B, k_C:

$$A \rightarrow B: \ (\{k_1\}_{k_B}, \{k_2\}_{k_B})$$
$$B \rightarrow C: \ (\{k_1\}_{k_C}, \{k_2\}_{k_1})$$

k_1, k_2: Local session keys.

- What will happen if one flips the order of messages in the first pair? It will produce $\{k_1\}_{k_2}$.

Motivating example

Consider the following protocol over A, B, C with public keys k_A, k_B, k_C:

$$A \to B: \ (\{k_1\}_{k_B}, \{k_2\}_{k_B})$$
$$B \to C: \ (\{k_1\}_{k_C}, \{k_2\}_{k_1})$$

k_1, k_2: Local session keys.

- What will happen if one flips the order of messages in the first pair? It will produce $\{k_1\}_{k_2}$.

  Conclusion-1: A key cycle may easily be produced in the presence of an active adversary.

Coinductive symbolic security

- We follow the general framework of Micciancio & Warinschi, but using co-induction (as opposed to induction) to model adversarial knowledge.

Coinductive symbolic security

- We follow the general framework of Micciancio & Warinschi, but using co-induction (as opposed to induction) to model adversarial knowledge.
- Coinduction was suggested by Miccinacio as tool to overcome limitations of previous soudnness theorems relying on the absence of key cycles.

Coinductive symbolic security

- We follow the general framework of Micciancio & Warinschi, but using co-induction (as opposed to induction) to model adversarial knowledge.
- Coinduction was suggested by Miccinacio as tool to overcome limitations of previous soudnness theorems relying on the absence of key cycles.
- Our work: applying co-induction in the case of active adversaries.

Computational soundness of coinductive symbolic security

- (Informal) For a protocol Π, a trace-expressible security property P, if all coinductive symbolic traces satisfy P (i.e., Π is coinductively secure), all (except a negligible fraction) of computational traces of any ARC-instantiation of Π against any PPT $\mathcal{A}$ satisfy P.
- Corollary (informal): If a protocol doesn't produce a "long" chain of key cycles, we can apply the soundness theorem to it (ie. Coinductive symbolic security implies computational security against adaptively corrupting adversaries)
- For all protocols that we considered from the Clark-Jacob library, the diameter of the corresponding coinductively-induced subgraph is at most 2, making the soundness theorem applicable to them.

Thanks!