Introduction to Security Reduction

Lecture 4

Entry to Security Reduction

Adversary

My IQ is up to 186.

My interest is breaking schemes.

You want me to help you solve problem?

Fool me first!

Lecture 12: Flaws in Papers
Lecture 11: Revision of Security Reduction
Lecture 10: Security Proofs for Encryption (Computational)
Lecture 9: Security Proofs for Encryption (Decisional)
Lecture 8: Security Proofs for Digital Signatures
Lecture 7: Analysis (Towards A Correct Reduction)
Lecture 6: Simulation and Solution
Lecture 5: Difficulties in Security Reduction
Lecture 4: Entry to Security Reduction
Lecture 3: Preliminaries (Hard Problem and Secure Scheme)
Lecture 2: Preliminaries (Field, Group, Pairing, and Hash Function)
Lecture 1: Definitions (Algorithm and Security Model)

Computational Complexity Theory

Outline

1 Reduction in Computational Complexity

2 Security Reduction in Cryptography
- Overview
- Framework
- Breaking to Solving

3 Evaluation of Security Reduction
- Cost and Loss in Reduction
- Concrete Security
- Ideal Security Reduction

Outline

1 Reduction in Computational Complexity

2 Security Reduction in Cryptography
- Overview
- Framework
- Breaking to Solving

3 Evaluation of Security Reduction
- Cost and Loss in Reduction
- Concrete Security
- Ideal Security Reduction

Proof by Contradiction

In computational complexity, without additional assumptions, it is "impossible" to prove that solving a computing problem B is hard.

In computational complexity, we can prove hardness of problem B with additional assumptions that some computing problems are hard.

The proof process is called reduction and it is a proof by contradiction.

> A computing problem A is believed to be *hard*.
> If problem B is *easy*, we prove that problem A is also *easy*.
> The assumption is then false, and the problem B must be *hard*.

We say:

The old problem A is reducible to the new problem B.

Reducible

In computational complexity,

The problem A is reducible to the problem B

means that

- Solving problem B can be transformed to solving problem A.
- We can transform a problem instance x_A into a problem instance x_B of problem B.
- We can transform the problem solution y_B to problem solution y_A.
- Problem B (the problem we care) is not easier than problem A.

Example (1)

Problem A: $x_A = (g, g^a, g^b), \quad y_A = g^{ab}$

Problem B: $x_B = (g, g^c, g^d), \quad y_B = g^{c^2+cd}$

We have:

The problem A is reducible to the problem B

Question: Do you know how to prove this?

Example (1)

Answer is given in the next page.

Example (1)

Problem A: $x_A = (g, g^a, g^b), \quad y_A = g^{ab}$

Problem B: $x_B = (g, g^c, g^d), \quad y_b = g^{c^2+cd}$

Proof. Given the instance (g, g^a, g^b), we work as follows.

1. Compute $\frac{g^b}{g^a} = g^{b-a}$.
2. Set $g^c = g^a$
3. Set $g^d = g^{b-a}$
4. Send the problem instance $(g, g^a, g^{b-a}) = (g, g^c, g^d)$ to algorithm $\mathcal{B}$ that can compute its problem solution g^{c^2+cd}.

We have

$$g^{c^2+cd} = g^{a^2+a(b-a)} = g^{a^2+ab-a^2} = g^{ab},$$

which is the solution to problem A. This completes the proof. □

Example (2)

Problem A: $x_A = (g, g^a, g^b), \quad y_A = g^{ab}$

Problem B: $x_B = (g, g^c, g^d), \quad y_B = g^{c^2 - d^2}$

We have:

The problem A is reducible to the problem B

Question: Do you know how to prove this?

No answer is given but the idea is quite similar.

Example (3)

Problem A: $x_A = (g, g^a, g^{a^2}, \cdots, g^{a^q}), \quad y_A = (c, g^{\frac{1}{a+c}})$ for any $c \in \mathbb{Z}_p$.

Problem B: $x_B = (g, h^{\frac{1}{a}}, h, h^a, \cdots, h^{a^{q-1}}), \quad y_A = (c, g^{\frac{1}{a+c}})$ for any $c \in \mathbb{Z}_p$.

We have:

The problem A is reducible to the problem B

Question: Do you know how to prove this?

Example (3)

Answer is given in the next page.

Example (3)

Problem A: $x_A = (g, g^a, g^{a^2}, \cdots, g^{a^q})$, $y_A = (c, g^{\frac{1}{a+c}})$ for any $c \in \mathbb{Z}_p$.

Problem B: $x_B = (g, h^{\frac{1}{a}}, h, h^a, \cdots, h^{a^{q-1}})$, $y_A = (c, g^{\frac{1}{a+c}})$ for any $c \in \mathbb{Z}_p$.

Proof. Given the instance $(g, g^a, g^{a^2}, \cdots, g^{a^q})$, we work as follows.

1. Randomly choose $w \in \mathbb{Z}_p$ and set $h = (g^a)^w$.
2. Send the problem instance

$$(g, h^{\frac{1}{a}}, h, h^a, \cdots, h^{a^{q-1}}) = (g, g^w, g^{wa}, \cdots, g^{wa^q})$$

   to algorithm $\mathcal{B}$ that can compute its problem solution $(c, g^{\frac{1}{a+c}})$.
3. We have the output is also the solution to problem A.

This completes the proof. □

Outline

1 Reduction in Computational Complexity

2 Security Reduction in Cryptography
- Overview
- Framework
- Breaking to Solving

3 Evaluation of Security Reduction
- Cost and Loss in Reduction
- Concrete Security
- Ideal Security Reduction

Outline

1 Reduction in Computational Complexity

2 Security Reduction in Cryptography
- Overview
- Framework
- Breaking to Solving

3 Evaluation of Security Reduction
- Cost and Loss in Reduction
- Concrete Security
- Ideal Security Reduction

Security Reduction (Overview)

Security reduction is similar to reduction in computational complexity.

Reduction	Problem	Reducible to	Problem
Security Reduction	Problem	Reducible to	Scheme

We prove security by proof by contradiction as follows.

> A computing problem P is believed to be *hard*.
> If a proposed scheme is *insecure*, we prove that the problem P is *easy*.
> The assumption is then false, and the scheme must be *secure*.

Note: The above is the high level of security reduction.

Security Reduction (Proof Strategy)

Recalling that

problem A is reducible to problem B

$\Downarrow$

problem instance x_A can be used to create problem instance x_B

Security Reduction (Proof Strategy)

Recalling that

problem A is reducible to problem B

$\Downarrow$

problem instance x_A can be used to create problem instance x_B

In security reduction,

a problem P is reducible to proposed scheme

$\Downarrow$

problem instance x_P can be used to "simulate" a scheme

$\Downarrow$

breaking the simulated scheme can be transformed to solution y_P

Outline

1 Reduction in Computational Complexity

2 Security Reduction in Cryptography
- Overview
- Framework
- Breaking to Solving

3 Evaluation of Security Reduction
- Cost and Loss in Reduction
- Concrete Security
- Ideal Security Reduction

Framework of Security Reduction

A security reduction must be carried out with the following framework.

Breaking Assumption: Suppose there exists an adversary who can break the proposed scheme (in a security model).

- **Simulation.** generates a simulated scheme (using problem instance x_P) and interacts with the adversary by following the security model.
- **Solution.** extracts problem solution y_P with the help of the adversary's attack on the simulated scheme.
- **Analysis.** show that the advantage of solving the hard problem P is non-negligible if the breaking assumption is true.

Analysis in Security Reduction

Breaking Assumption: Suppose there exists an adversary who can break the proposed scheme (in a security model).

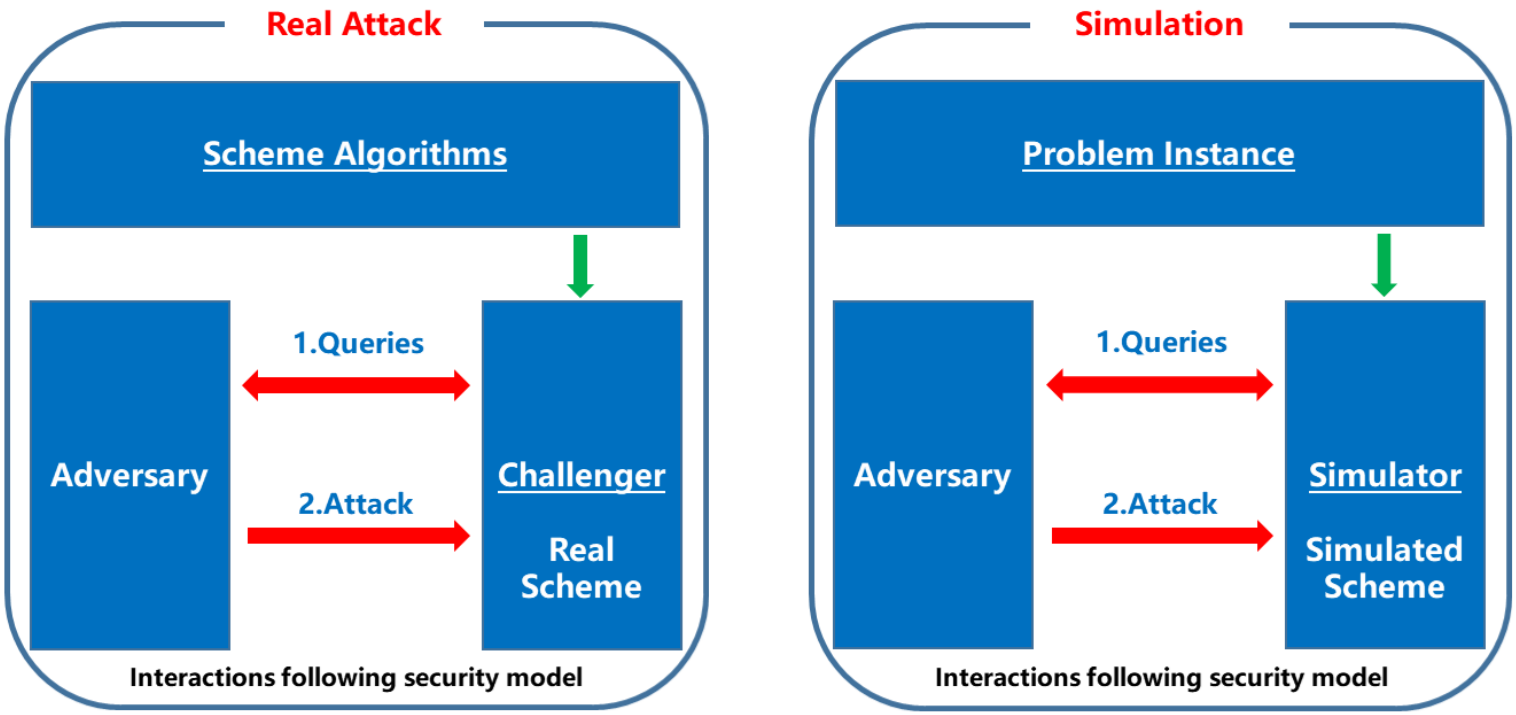
- **Simulation.** generates a simulated scheme (using problem instance x_P) and interacts with the adversary by following the security model.
- **Solution.** extracts problem solution y_P with the help of the adversary's attack on the simulated scheme.

Question: Can we program a security reduction without analysis?

Analysis in Security Reduction

- A security reduction is not a real mathematical proof.
- Instead, it merely proposes a reduction algorithm and shows how to reduce the adversary's attack to solving a hard problem.
- That is, a security reduction is a reduction algorithm only.
- Unfortunately, we cannot demonstrate this reduction algorithm to convince people that the reduction algorithm works because there is no adversary.
- What we do instead is a theoretical analysis showing that the proposed reduction algorithm indeed works.

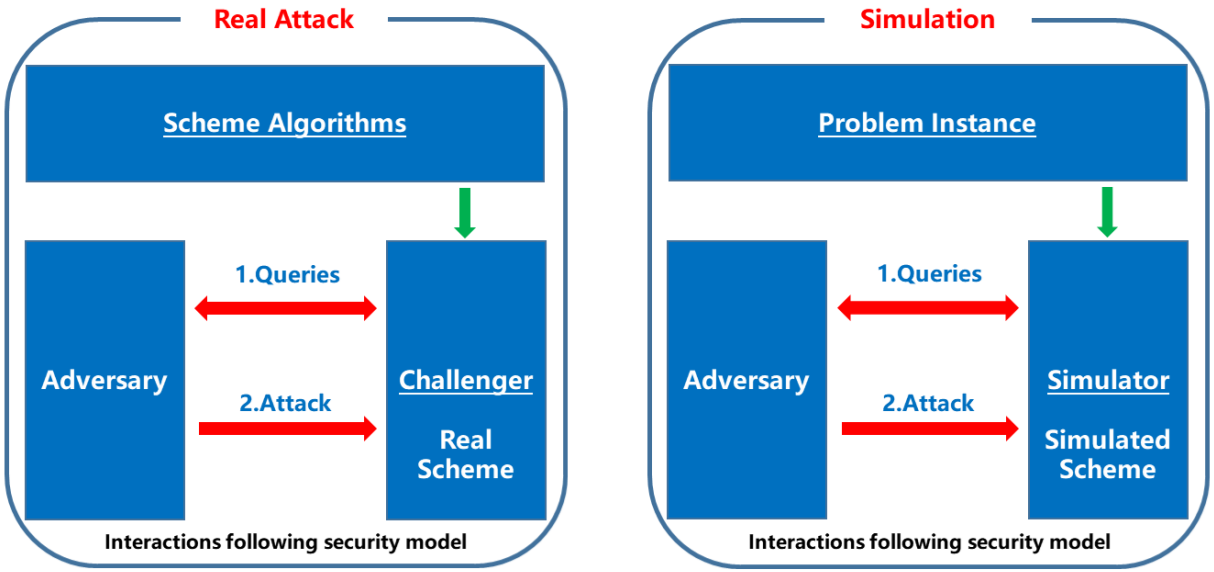
Concepts (1)



Concepts in real attacks and simulation.

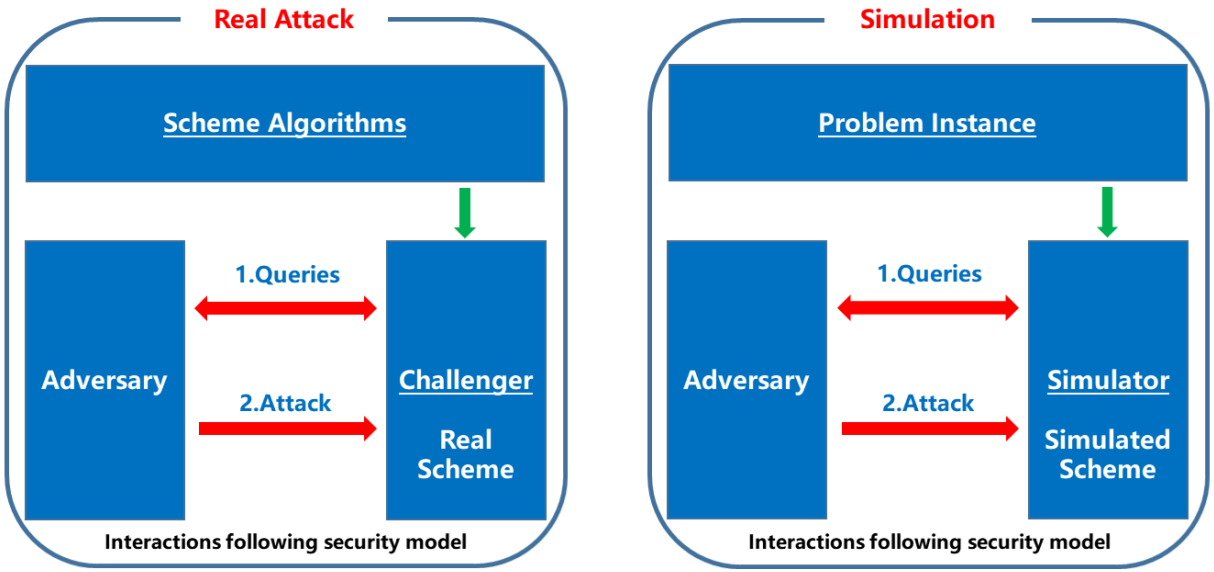
Real Scheme	vs	Simulated Scheme
Challenger	vs	Simulator
Real attack	vs	Simulation

Concepts (2)



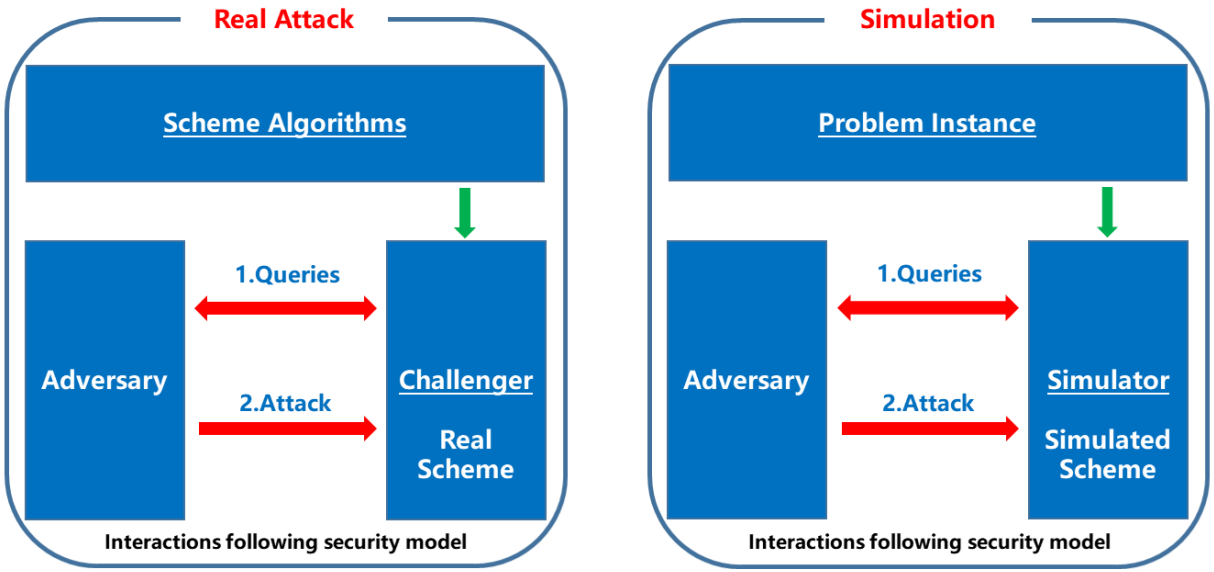
- A real scheme is a scheme generated with a security parameter following the scheme algorithm described in the proposed scheme.
- A simulated scheme is a scheme generated with a random instance of an underlying hard problem following the reduction algorithm.

Concepts (3)



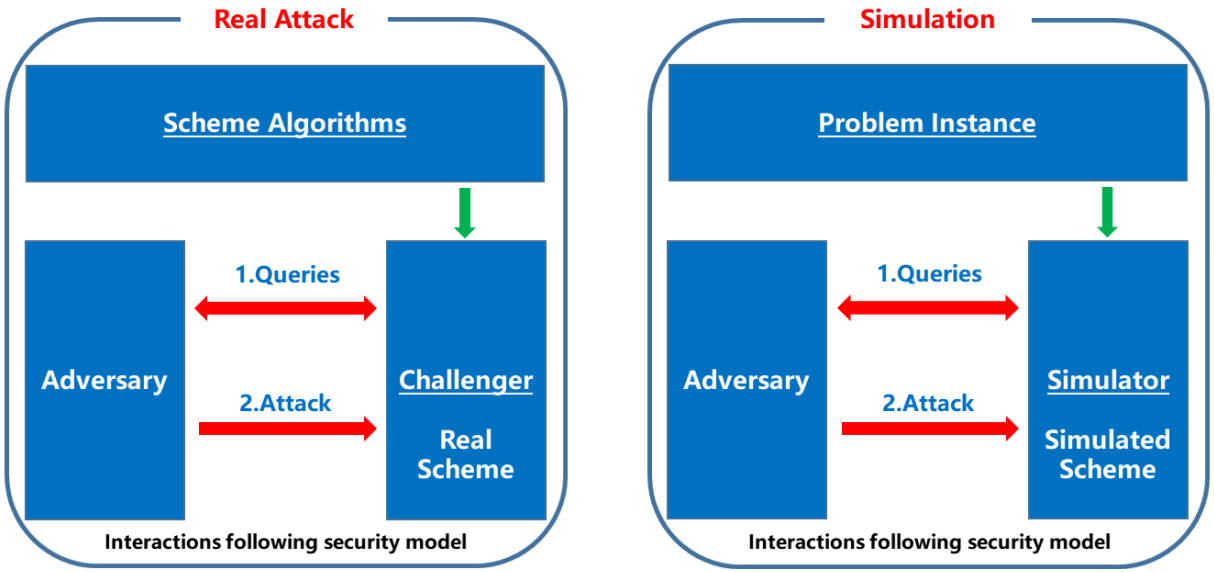
- The adversary interacts with real scheme controlled by the challenger. The challenger appears in the security model.
- The adversary interacts with simulated scheme controlled by the simulator. The simulator appears in the security reduction.

Concepts (4)



- The real attack is the interaction between the adversary and the challenger. (The information that the adversary knows.)
- The simulation is the interaction between the adversary and the simulator.(The information that the adversary knows.)

Concepts (5)



- From the view of the adversary, the responses by real scheme and simulated scheme could be indistinguishable (looks the same).
- The responses by real scheme is following scheme algorithm and the response by simulated scheme is following reduction algorithm.

Indistinguishable: First View

Approach 1:

Randomly chooses a from $\{0, 1, \cdots, 9\}$ and computes $b = a + 1 \mod 10$

Approach 2:

Randomly chooses a from $\{0, 1, \cdots, 9\}$ and computes $b = a + 7 \mod 10$

Question: Given an integer b, can we know it is computed from Approach 1 or Approach 2 ?

Outline

1 Reduction in Computational Complexity

2 Security Reduction in Cryptography
- Overview
- Framework
- Breaking to Solving

3 Evaluation of Security Reduction
- Cost and Loss in Reduction
- Concrete Security
- Ideal Security Reduction

Solution

The aim of a security reduction is to reduce an adversary's attack to solving an underlying hard problem. An attack can be a computational attack or a decisional attack.

- A computational attack, such as forging a valid signature, requires the adversary to find a correct answer from an exponential-size answer space.
- A decisional attack, such as guessing the message $m_b \in \{m_0, m_1\}$ in the challenge ciphertext in the IND-CPA security model, only requires the adversary to guess b (0 or 1).

Note: Security against a decisional attack = Indistinguishability security.

Three Types

	Attack		Hard Problem
Type 1	Computational Attacks	$\rightarrow$	Computational Hard Problems
Type 2	Decisional Attacks	$\rightarrow$	Decisional Hard Problems
Type 3	Decisional Attacks	$\rightarrow$	Computational Hard Problems

Note: The third type is very special because it is only available in the random oracle model, where the simulator uses hash queries made by the adversary to solve a computational hard problem.

Outline

1 Reduction in Computational Complexity

2 Security Reduction in Cryptography
- Overview
- Framework
- Breaking to Solving

3 Evaluation of Security Reduction
- Cost and Loss in Reduction
- Concrete Security
- Ideal Security Reduction

Outline

1 Reduction in Computational Complexity

2 Security Reduction in Cryptography
- Overview
- Framework
- Breaking to Solving

3 **Evaluation of Security Reduction**
- Cost and Loss in Reduction
- Concrete Security
- Ideal Security Reduction

Reduction Cost and Reduction Loss

Suppose there exists an adversary who can (t, ϵ)-break the scheme.

Generally speaking, we solve an underlying hard problem with

$$t' = t + T, \quad \epsilon' = \frac{\epsilon}{L}.$$

- T is *reduction cost* (simulation time). We have

$$T = O(q)$$

- L is *reduction loss* (success probability $\frac{1}{L}$). We have

$$1 \leq L \leq q$$

Here, q is the number of queries, such as signature queries or hash queries or private key queries.

Tight Reduction and Loose Reduction

Suppose we solve an underlying hard problem with

$$t' = t + T, \quad \epsilon' = \frac{\epsilon}{L}.$$

- The security reduction is a tight reduction if

$$L << q,$$

  where L can be such as $L = O(1), L = O(\log^q)$.

- The security reduction is a loose reduction if

$$L = O(q)$$

Note: We rarely consider the simulation time in tight reductions.

Tight Reduction: Necessary or Not?

- A tight reduction is better than a loose reduction.
- How to program a tight reduction is a hot research topic in the theory of cryptography research.
- Applied cryptography doesn't care loose reduction too much.
- A scheme with tight reduction is usually less efficient in computations than a scheme with a loose reduction.
- Finding a scheme with a high computational efficiency requirement shouldn't consider tight reduction.

Outline

1 Reduction in Computational Complexity

2 Security Reduction in Cryptography
- Overview
- Framework
- Breaking to Solving

3 Evaluation of Security Reduction
- Cost and Loss in Reduction
- Concrete Security
- Ideal Security Reduction

Lower Bound Security Level Revisited

- Suppose breaking the scheme S can be reduced to solving an underlying hard problem, denoted by B.
- The lower bound security level of the scheme S is calculated from the underlying hard problem B.
- However,

  Lower bound security level of $S \neq$ Security level of the problem B.

It still depends on the reduction cost and the reduction loss.

Concrete Security

Suppose the hard problem B has k-bit security.
Suppose the scheme S can be broken with (t, ϵ).
Suppose a reduction solves the hard problem B with

$$t' = t + T, \quad \epsilon' = \frac{\epsilon}{L}.$$

We have

$$\frac{t+T}{\frac{\epsilon}{L}} \geq 2^k$$

We obtain

$$\frac{t}{\epsilon} \geq 2^{k-\log^L} - \frac{T}{\epsilon}.$$

which is the concrete security of scheme via formal reduction.

Concrete Security

$$\frac{t}{\epsilon} \geq 2^{k-\log^L} - \frac{T}{\epsilon}.$$

- The lower bound security level of scheme S will be very low if L and T are high.
- The lower bound security level becomes high if we can find a better security reduction with small T and L.

That is, with security reduction, we prove that the concrete security of scheme is at least (for example) $2^{\lambda-30}$. We cannot say that the concrete security is $2^{\lambda-30}$.

Outline

1 Reduction in Computational Complexity

2 Security Reduction in Cryptography
- Overview
- Framework
- Breaking to Solving

3 Evaluation of Security Reduction
- Cost and Loss in Reduction
- Concrete Security
- Ideal Security Reduction

Ideal Security Reduction

An ideal security reduction is the best security reduction that we can program for a proposed scheme.

- **Security Model.** The security model that allows the adversary to maximally, flexibly, and adaptively make queries to the challenger and win the game with a minimum requirement.
- **Hard Problem.** The underlying hard problem adopted must be the hardest one among all hard problems defined over the same mathematical primitive. For example, the DL problem.
- **Reduction Cost and Reduction Loss.** The reduction cost T and the reduction loss L are the minimized values. That is, T is linear in the number of queries made by the adversary and $L = 1$.
- **Computational Restrictions on Adversary.** There is no computational restriction on the adversary except time and advantage. The random oracle model does restrict the adversary.

Ideal Security Reduction

Unfortunately, an inherent tradeoff among these ideal features is very common in all security reductions proposed in the literature.

- For example, we can construct an efficient signature scheme whose security is under a weak hardness assumption, but the security reduction must use random oracles.
- For example, we can also construct a signature scheme without random oracles in the security reduction, but it is accompanied with a strong assumption or a long public key.

Currently, it seems technically impossible to construct a scheme with an ideal security reduction satisfying all four features mentioned above.

Let us create a human-like AI Robot with one line code only!


END