Introduction to Security Reduction

Lecture 5

Difficulties in Security Reduction

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

Introduction to Security Reduction

Outline

1 Understanding Security Reduction

2 Simulation and After Simulation
- Something Unknown
- Simulation Results
- Classification of Attacks
- Relations

3 Adversary
- From Black-Box to Malicious
- Adversary's Computing Ability
- What Adversary Knows and Never Knows
- Summary of Adversary

Outline

1 **Understanding Security Reduction**

2 Simulation and After Simulation
- Something Unknown
- Simulation Results
- Classification of Attacks
- Relations

3 Adversary
- From Black-Box to Malicious
- Adversary's Computing Ability
- What Adversary Knows and Never Knows
- Summary of Adversary

Introduction to Security Reduction

Understanding Security Reduction

Proof. Suppose there exists an adversary who can break the proposed scheme (in a security model). Given a problem instance x_P, we can construct a simulator to generate a simulated scheme and use the adversary's attack on the simulated scheme to compute y_P.

$\vdots$

- The adversary is able to break the proposed (real) scheme.
- The adversary is asked to break the simulated scheme.

Understanding Security Reduction

Proof. Suppose there exists an adversary who can break the proposed scheme (in a security model). Given a problem instance x_P, we can construct a simulator to generate a simulated scheme and use the adversary's attack on the simulated scheme to compute y_P.

⋮

- The adversary is able to break the proposed (real) scheme.
- The adversary is asked to break the simulated scheme.

Silly, honey bee likes flowers but not the plastic one!

Understanding Security Reduction

The adversary is attacking a given scheme $\in$ {real scheme, simulated scheme}.

- The adversary is assumed to be able to break any given scheme following the security model if the given scheme looks like a real scheme from the point of view of the adversary.
- The adversary interacts with a given scheme that is a simulated scheme in security reduction. We want the adversary to believe that the given scheme is a real scheme and break it.
- We don't know whether the adversary will break the simulated scheme with the same advantage as breaking the real scheme if the adversary finds out that the given scheme is not a real scheme.
- We also don't know how the adversary breaks the simulated scheme when the given scheme looks like a real scheme.

Understanding 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!

Outline

1 Understanding Security Reduction

2 Simulation and After Simulation
- Something Unknown
- Simulation Results
- Classification of Attacks
- Relations

3 Adversary
- From Black-Box to Malicious
- Adversary's Computing Ability
- What Adversary Knows and Never Knows
- Summary of Adversary

Introduction to Security Reduction

Outline

1 Understanding Security Reduction

2 Simulation and After Simulation
- Something Unknown
- Simulation Results
- Classification of Attacks
- Relations

3 Adversary
- From Black-Box to Malicious
- Adversary's Computing Ability
- What Adversary Knows and Never Knows
- Summary of Adversary

Real Scheme VS Simulated Scheme

- In the real scheme, the challenger generates the scheme and knows everything including the secret key.
- In the simulated scheme, the simulator generates the scheme with a problem instance and doesn't know something.

For example, a simulated scheme is generated with a problem instance (g, g^a, g^b) and the simulator therefore doesn't know (a, b) in simulation.

The unknown value might affect the simulation. If simulator knows everything, why does he still need the adversary's help?

Outline

1 Understanding Security Reduction

2 Simulation and After Simulation
- Something Unknown
- Simulation Results
- Classification of Attacks
- Relations

3 Adversary
- From Black-Box to Malicious
- Adversary’s Computing Ability
- What Adversary Knows and Never Knows
- Summary of Adversary

Successful Simulation

A simulation is successful if the simulator can respond to all queries from the adversary following a security model.

- The simulator doesn't abort in the simulation.
- The simulator makes the decision to abort the simulation or not according to the reduction algorithm.

For example, the simulator aborts because it doesn't know how to generate a queried signature for the adversary.

Indistinguishable Simulation

A simulation is indistinguishable from real attack if the adversary cannot distinguish the simulated scheme from the real scheme.

A successful simulation is an indistinguishable simulation if

- Correctness. Responses to queries following security model in simulated scheme are correct. For example, a simulated signature on m must be valid and can pass signature verification.
- Randomness. Random numbers in simulated scheme are random and independent. All simulated random numbers are random and independent the same as real random numbers.

Note: The adversary can only try to distinguish simulated scheme from real scheme via correctness or randomness. For example, we don't consider the difference of time cost in query response.

Outline

1 Understanding Security Reduction

2 Simulation and After Simulation
- Something Unknown
- Simulation Results
- Classification of Attacks
- Relations

3 Adversary
- From Black-Box to Malicious
- Adversary's Computing Ability
- What Adversary Knows and Never Knows
- Summary of Adversary

Attacks from the Adversary

According to results of attack on simulated scheme, we have

- **Failed Attack.** An attack fails if the attack cannot break the simulated scheme following the security model. For example, the forged signature is invalid.
- **Successful Attack.** An attack is successful if the attack can break the simulated scheme following the security model. For example, the forged signature is valid.

According to results of solving hard problem, we have

- **Useless Attack.** A useless attack is an attack by the adversary that cannot be reduced to solving an underlying hard problem.
- **Useful Attack.** A useful attack is an attack by the adversary that can be reduced to solving an underlying hard problem.

Introduction to Security Reduction

Example

We define a real attack as follows.

- The adversary is given $(g, g^a, g^{b_1}, g^{b_2}, g^{b_3})$ generated by the challenger who knows (b_1, b_2, b_3).
- The adversary can adaptive query an integer $j \in \{1, 2, 3\}$ and the challenger responds with g^{ab_j}.
- The adversary breaks the scheme if it can compute $g^{ab_{i^*}}$ satisfying:

$$i^* \in \{1, 2, 3\} \wedge i^* \neq j.$$

Reduction: We can reduce adversary's attack to solving CDH problem.

Example

Answer is given in the next page.

Example

Proof. Given (g, g^a, g^b), the simulator works as follows.

- Randomly choose $(c_1, c_2, c_3) \in \mathbb{Z}_p$ and $k \in \{1, 2, 3\}$. Set

$$b_i = \begin{cases} c_i & \text{if } i \neq k \\ b & \text{if } i = k \end{cases}$$

- We have $(g^{b_1}, g^{b_2}, g^{b_3})$ can be simulated from (c_1, c_2, c_3, g^b).
- Send $(g, g^a, g^{b_1}, g^{b_2}, g^{b_3})$ to the adversary.
- The adversary makes an adaptive choice j. The simulator aborts if $j = k$. Otherwise, we have $b_j = c_j$ and the simulator can compute $g^{ab_j} = (g^a)^{c_j}$ for the adversary.
- The adversary outputs $g^{ab_{i^*}}$. We have $g^{ab_{i^*}} = g^{ab}$ if $i^* = k$.

This completes the simulation and the solution. □

Example

Proof. Given (g, g^a, g^b), the simulator works as follows.

- Randomly choose $(c_1, c_2, c_3) \in \mathbb{Z}_p$ and $k \in \{1, 2, 3\}$. Set

$$b_i = \begin{cases} c_i & \text{if } i \neq k \\ b & \text{if } i = k \end{cases}$$

- We have $(g^{b_1}, g^{b_2}, g^{b_3})$ can be simulated from (c_1, c_2, c_3, g^b).
- Send $(g, g^a, g^{b_1}, g^{b_2}, g^{b_3})$ to the adversary.
- The adversary makes an adaptive choice j. The simulator aborts if $j = k$. Otherwise, we have $b_j = c_j$ and the simulator can compute $g^{ab_j} = (g^a)^{c_j}$ for the adversary.
- The adversary outputs $g^{ab_{i^*}}$. We have $g^{ab_{i^*}} = g^{ab}$ if $i^* = k$.

The simulation is successful if the query satisfies $j \neq k$.

Example

Proof. Given (g, g^a, g^b), the simulator works as follows.

- Randomly choose $(c_1, c_2, c_3) \in \mathbb{Z}_p$ and $k \in \{1, 2, 3\}$. Set

$$b_i = \begin{cases} c_i & \text{if } i \neq k \\ b & \text{if } i = k \end{cases}$$

- We have $(g^{b_1}, g^{b_2}, g^{b_3})$ can be simulated from (c_1, c_2, c_3, g^b).
- Send $(g, g^a, g^{b_1}, g^{b_2}, g^{b_3})$ to the adversary.
- The adversary makes an adaptive choice j. The simulator aborts if $j = k$. Otherwise, we have $b_j = c_j$ and the simulator can compute $g^{ab_j} = (g^a)^{c_j}$ for the adversary.
- The adversary outputs $g^{ab_{i^*}}$. We have $g^{ab_{i^*}} = g^{ab}$ if $i^* = k$.

The attack is a successful attack if the adversary can output $g^{ab_{i^*}}$.

Example

Proof. Given (g, g^a, g^b), the simulator works as follows.

- Randomly choose $(c_1, c_2, c_3) \in \mathbb{Z}_p$ and $k \in \{1, 2, 3\}$. Set

$$b_i = \begin{cases} c_i & \text{if } i \neq k \\ b & \text{if } i = k \end{cases}$$

- We have $(g^{b_1}, g^{b_2}, g^{b_3})$ can be simulated from (c_1, c_2, c_3, g^b).
- Send $(g, g^a, g^{b_1}, g^{b_2}, g^{b_3})$ to the adversary.
- The adversary makes an adaptive choice j. The simulator aborts if $j = k$. Otherwise, we have $b_j = c_j$ and the simulator can compute $g^{ab_j} = (g^a)^{c_j}$ for the adversary.
- The adversary outputs $g^{ab_{i^*}}$. We have $g^{ab_{i^*}} = g^{ab}$ if $i^* = k$.

The attack is useless when $i^* \neq k$, and useful when $i^* = k$.

Introduction to Security Reduction

Outline

1 Understanding Security Reduction

2 Simulation and After Simulation
- Something Unknown
- Simulation Results
- Classification of Attacks
- Relations

3 Adversary
- From Black-Box to Malicious
- Adversary's Computing Ability
- What Adversary Knows and Never Knows
- Summary of Adversary

Logic in Security Reduction

We create a simulated scheme with a problem instance.

$\Downarrow$

Successful Simulation

$\Downarrow$

Indistinguishable Simulation

$\Downarrow$

Successful Attack

$\Downarrow$

Useful Attack

$\Downarrow$

Solve Hard Problem

Security reduction is complicated because this is not the only logic!

Relations

- An attack at the end of a successful simulation can be either a failed attack or a successful attack, which is dependent on whether the simulation is distinguishable or not.
- An attack at the end of a distinguishable simulation can be either a failed attack or a successful attack. That is, the adversary can decide which attack it will launch. This doesn't contradict the breaking assumption.
- An attack at the end of an indistinguishable simulation must be a successful attack with probability defined in the breaking assumption.
- The attack on the indistinguishable simulated scheme can be either a useful attack or a useless attack. An indistinguishable simulation cannot ensure a useful attack.

Relations

We might receive a useful attack from the adversary in some very special security reductions, even

- The simulated scheme is distinguishable from real scheme, or
- The attack from the adversary is a failed attack.

We might receive a useless attack from the adversary in many security reductions, even

- The simulated scheme is indistinguishable from real scheme, and
- The attack from the adversary is a successful attack.

Unfortunately, no example or further explanation now. :(

Relations

- **Successful Security Reduction.** We say that a security reduction is successful if the adversary outputs an attack and the attack is a useful attack.
- **Correct Security Reduction.** We say that a security reduction for a proposed scheme is correct if the advantage of solving an underlying hard problem using the adversary's attack is non-negligible in polynomial time.

Note: A successful reduction can be treated as a good event of security reduction. We need to make sure that the good event occurs with a non-negligible probability (for obtaining a correct reduction).

Outline

1 Understanding Security Reduction

2 Simulation and After Simulation
- Something Unknown
- Simulation Results
- Classification of Attacks
- Relations

3 Adversary
- From Black-Box to Malicious
- Adversary's Computing Ability
- What Adversary Knows and Never Knows
- Summary of Adversary

I know who you (simulator) are!

You want to fool me?

I will fool you first!

My IQ is up to 186......

Outline

1 Understanding Security Reduction

2 Simulation and After Simulation
- Something Unknown
- Simulation Results
- Classification of Attacks
- Relations

3 Adversary
- From Black-Box to Malicious
- Adversary's Computing Ability
- What Adversary Knows and Never Knows
- Summary of Adversary

Black-box Adversary

Suppose there exists an adversary who can (t, ϵ)-break the proposed scheme in a security model. We construct a simulator to solve a hard problem.

- There is no restriction on the adversary except time and advantage.
- We don't know how the adversary breaks the scheme except that the attack on real scheme works, even the attack looks impossible to be launched from the view of simulator.
- For example, the adversary needs to solve some hard problems.

The adversary in security reduction is a black-box adversary.

Black-box Adversary

The output from a black-box adversary is not uniformally distributed but

Adaptive

Let a be an integer chosen from the set $\{0, 1\}$.

- If a is randomly chosen, we have

$$\Pr[a = 0] = \Pr[a = 1] = \frac{1}{2}.$$

- If a is adaptively chosen, we have

$$\Pr[a = 0], \Pr[a = 1] : \text{unknown}.$$

Note: **Adaptive = Unpredictable** $\neq$ **Adaptive** Chosen-Message Attacks

Black-box Adversary

The adaptive output from the black-box adversary includes

- Adaptive query.
- Adaptive attack.

Example. Suppose the message space is $\{m_1, m_2, m_3, m_4, m_5\}$ with five messages. An adversary will query the signature of one message before it forges a valid signature of a new message.

- Adaptive query. The adversary will query the signature of message m_i with unknown probability.
- Adaptive attack. If the adversary makes a signature query on m_4, the adversary will forge a signature of one of the messages from $\{m_1, m_2, m_3, m_5\}$ with unknown probabilities between $[0, 1]$ satisfying

$$\Pr[m^* = m_1] + \Pr[m^* = m_2] + \Pr[m^* = m_3] + \Pr[m^* = m_5] = 1.$$

Black-box Adversary to Malicious Adversary

In security reduction,

- We construct a simulator to "fool" any black-box adversary in helping us solve a hard problem.
- If we cannot fool one particular adversary, we fail!
- We must be able fool the most smart/cunning adversary.

Black-box Adversary to Malicious Adversary

In security reduction,

- We construct a simulator to "fool" any black-box adversary in helping us solve a hard problem.
- If we cannot fool one particular adversary, we fail!
- We must be able fool the most smart/cunning adversary.

Suppose there are only two distinct attacks that can break the simulated scheme in a security reduction. One attack is useful and the other is useless. Consider the question.

What is the probability of returning a useful attack by the adversary?

Black-box Adversary to Malicious Adversary

What is the probability of returning a useful attack by the adversary?

- This probability is unknown due to the adaptive attack by the black-box adversary.
- However, a correct security reduction requires us to calculate the probability of returning a useful attack (solving hard problem).
- We amplify the black-box adversary into a malicious adversary and consider the *maximum probability of returning a useless attack* by the malicious adversary.

Black-box Adversary to Malicious Adversary

- The malicious adversary will try its best to launch a useless attack unless the adversary doesn't know which attack is useless.
- If the maximum probability of returning a useless attack is not the overwhelming probability 1, the probability of returning a useful attack must be noticeable (non-negligible).
- If a security reduction works against such a malicious adversary, the security reduction definitely works against any adversary who can break the proposed scheme.

In short, we need to calculate probability but black-box adversary doesn't allow us to do this. We therefore amplify it into malicious one.

Example

Proof. Given (g, g^a, g^b), the simulator works as follows.

- Randomly choose $(c_1, c_2, c_3) \in \mathbb{Z}_p$ and $k \in \{1, 2, 3\}$. Set

$$b_i = \begin{cases} c_i & \text{if } i \neq k \\ b & \text{if } i = k \end{cases}$$

- We have $(g^{b_1}, g^{b_2}, g^{b_3})$ can be simulated from (c_1, c_2, c_3, g^b).
- Send $(g, g^a, g^{b_1}, g^{b_2}, g^{b_3})$ to the adversary.
- The adversary makes an adaptive choice j. The simulator aborts if $j = k$. Otherwise, we have $b_j = c_j$ and the simulator can compute $g^{ab_j} = (g^a)^{c_j}$ for the adversary.
- The adversary outputs $g^{ab_{i^*}}$. We have $g^{ab_{i^*}} = g^{ab}$ if $i^* = k$.

In this reduction, the malicious adversary will try its best to guess k before it outputs $g^{ab_{i^*}}$ (try to launch a useless attack).

Malicious Adversary and Useful Attack

Let P_ϵ be the probability of returning a successful attack on the proposed scheme under the breaking assumption.

- If the simulated scheme is indistinguishable from the real scheme, according to the breaking assumption, the adversary will return a successful attack on the simulated scheme with probability P_ϵ.
- If the simulated scheme is distinguishable from the real scheme, the adversary will return a successful attack on the simulated scheme with malicious and adaptive probability $P^* \in [0, 1]$.

Note: Returning a successful attack with malicious and unknown probability $P^* \in [0, 1]$ doesn't contradict the breaking assumption, because the simulated scheme is different from the real scheme.

Outline

1 Understanding Security Reduction

2 Simulation and After Simulation
- Something Unknown
- Simulation Results
- Classification of Attacks
- Relations

3 **Adversary**
- From Black-Box to Malicious
- **Adversary’s Computing Ability**
- What Adversary Knows and Never Knows
- Summary of Adversary

Introduction to Security Reduction

Adversary’s Computing Ability

Theorem
If the mathematical problem P is hard, the proposed scheme is secure and there exists no adversary who can break the proposed scheme in polynomial time with non-negligible advantage.

- We only consider whether the proposed scheme is secure or not when problem P is hard.
- We don’t care that the proposed scheme is insecure when solving problem P is easy.

Question: Can the adversary solve the problem P in reduction?

Adversary's Computing Ability

- Security reduction is to obtain the result that the proposed scheme is secure against any adversary who cannot solve the hard problem P in polynomial time.
- However, in security reduction, we don't care whether the proposed scheme can be broken or not.
- Instead, we only care whether the hard problem can be solved or not in polynomial time.

It is okay even we amplify the adversary's computational ability in security reduction, right?

Adversary's Computing Ability

- The proposed scheme must be computationally secure only.
- We program reduction in a way that solving hard problems also works even the adversary is computationally unbounded.
- The computationally unbounded adversary can solve all computationally hard problems in polynomial time with advantage 1.

Benefit: The amplification will simplify the analysis of reduction.

Dis-Benefit: Programming a correct security reduction becomes harder.

Benefits of Amplification

- The simulation in security reduction, such as response to signature query, might carry some secret information (denoted by $\mathbb{I}$).
- This piece of information $\mathbb{I}$ tells the adversary how to generate a useless attack on the simulated scheme.
- We need to analyze that the adversary cannot obtain $\mathbb{I}$.

Note: According to all proposed security reductions in the literature, it seems impossible not to include the information $\mathbb{I}$ in responses to the adversary.

Introduction to Security Reduction

Example of $\mathbb{I}$

Proof. Given (g, g^a, g^b), the simulator works as follows.

- Randomly choose $(c_1, c_2, c_3) \in \mathbb{Z}_p$ and $k \in \{1, 2, 3\}$. Set

$$b_i = \begin{cases} c_i & \text{if } i \neq k \\ b & \text{if } i = k \end{cases}$$

- We have $(g^{b_1}, g^{b_2}, g^{b_3})$ can be simulated from (c_1, c_2, c_3, g^b).
- Send $(g, g^a, g^{b_1}, g^{b_2}, g^{b_3})$ to the adversary.
- The adversary makes an adaptive choice j. The simulator aborts if $j = k$. Otherwise, we have $b_j = c_j$ and the simulator can compute $g^{ab_j} = (g^a)^{c_j}$ for the adversary.
- The adversary outputs $g^{ab_{i^*}}$. We have $g^{ab_{i^*}} = g^{ab}$ if $i^* = k$.

The elements $(g, g^a, g^{b_1}, g^{b_2}, g^{b_3})$ carries the information $\mathbb{I} = k$.

Example of Benefit

Prove that the adversary cannot compute a from the following elements:

$$\left(g, g^{x+z}, g^{x+a+y}, g^{z+a^2}\right),$$

where x, y, z are randomly chosen from $\mathbb{Z}_p$.

After the amplification, we only need to prove that the adversary cannot compute a from the following elements:

$$\left(x+z, \quad x+a+y, \quad z+a^2\right).$$

The algebra methodology will be applied in analysis (in later lectures).

Outline

1 Understanding Security Reduction

2 Simulation and After Simulation
- Something Unknown
- Simulation Results
- Classification of Attacks
- Relations

3 Adversary
- From Black-Box to Malicious
- Adversary's Computing Ability
- What Adversary Knows and Never Knows
- Summary of Adversary

Adversary in Reduction

A security reduction starts with breaking assumption. Then, we construct a simulator to generate a simulated scheme and use the adversary's attack to solve an underlying hard problem.

- The adversary has unbounded computational power in solving all computational hard problems.
- The adversary will maliciously try its best to launch a useless attack to break the simulated scheme and make the reduction fail.

Note: We have to use an attack from the above adversary to solve an underlying hard problem with non-negligible advantage.

What the Adversary Knows

There are three types of information that the adversary knows.

- **Scheme Algorithm.** The adversary knows the scheme algorithm of the proposed scheme, such as signing and verification algorithm.
- **Reduction Algorithm.** The adversary knows the reduction algorithm for proving the security of the proposed scheme. For example, the adversary knows how signatures are simulated.
- **How to Solve All Computational Hard Problems.** For example, suppose (g, g^a) is given to the adversary. We assume that the adversary can compute a before launching an attack.

What the Adversary Never Knows

There are three types of information that the adversary never knows.

- **Random Numbers.** The adversary doesn't know those random numbers (including group elements) chosen by the simulator unless they can be computed by the adversary.
- **Problem Instance.** The adversary doesn't know the random instance of the underlying hard problem given to the simulator.
- **How to Solve an Absolutely Hard Problem.** Such as computing (x, y) from the group elements (g, g^{x+y}). (This type of problems will be introduced later.)

Note: If the simulator randomly chooses $x, y \in \mathbb{Z}_p$, they are unknown to the adversary. However, once (g, g^{x+y}) are given to the adversary, the adversary can compute and know $x + y$.

Example of What the Adversary Knows

Proof. Given (g, g^a, g^b), the simulator works as follows.

- Randomly choose $(c_1, c_2, c_3) \in \mathbb{Z}_p$ and $k \in \{1, 2, 3\}$. Set

$$b_i = \begin{cases} c_i & \text{if } i \neq k \\ b & \text{if } i = k \end{cases}$$

- We have $(g^{b_1}, g^{b_2}, g^{b_3})$ can be simulated from (c_1, c_2, c_3, g^b).
- Send $(g, g^a, g^{b_1}, g^{b_2}, g^{b_3})$ to the adversary.
- The adversary makes an adaptive choice j. The simulator aborts if $j = k$. Otherwise, we have $b_j = c_j$ and the simulator can compute $g^{ab_j} = (g^a)^{c_j}$ for the adversary.
- The adversary outputs $g^{ab_{i^*}}$. We have $g^{ab_{i^*}} = g^{ab}$ if $i^* = k$.

The adversary knows a, b_1, b_2, b_3 before it launches an attack.

Outline

1 Understanding Security Reduction

2 Simulation and After Simulation
- Something Unknown
- Simulation Results
- Classification of Attacks
- Relations

3 Adversary
- From Black-Box to Malicious
- Adversary's Computing Ability
- What Adversary Knows and Never Knows
- Summary of Adversary

Summary (1)

- When the adversary is asked to interact with a given scheme. The adversary will use what it knows and what it can query (following security model) to find whether the given scheme is a simulated scheme.
- When the adversary finds out that the given scheme is a simulated scheme, the adversary will launch a successful attack with malicious and adaptive probability $P^* \in [0, 1]$.
- When the adversary cannot distinguish the simulation from the real attack, it will launch a successful attack with probability P_ϵ according to the breaking assumption.
- Assuming that the given scheme is a simulated scheme, the adversary will use what it knows and what it receives to launch a useless attack on the given scheme.

Summary (2)

- Security Reduction. Suppose there exists an adversary who can break the proposed scheme. We can construct a simulator to solve an underlying hard problem.
- Correct Security Reduction. Even if the attack on the simulated scheme is launched by a malicious adversary who has unbounded computational power, the advantage of solving the underlying hard problem is still non-negligible (in polynomial time).

Can you survive from my useless-attack fooling?

END