Introduction to Security Reduction

Lecture 11: Revision of 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

Topics in this Subject

Lecture 8-10 (Signature and Encryption)	
Lecture 6-7 (Simulation, Solution, Analysis)	
Lecture 5 (Difficulties in Security Reduction)	
Lecture 4 (Entry to Security Reduction)	
Lecture 3 (Hard Problem and Secure Scheme)	
Lecture 2 (Group, Pairing, Hash)	**Lecture 1** (Definition)

This lecture is going to revisit 50 questions.

Outline

1. Lecture 1-3: Scheme and Problem
2. Lecture 4-7: Difficulties and Approaches
3. Lecture 8-10: Signature and Encryption
4. Conclusion

Outline

Question 1: What should we consider when defining algorithms?

Question 1: What should we consider when defining algorithms?

- Fully understand the security service(s) (motivation).
- Which entities (e.g., signer) are involved.
- How many algorithms are involved.
- What is the name of each algorithm.
- Who will runs each algorithm.
- What are inputs and outputs of each algorithm (objects).
- What correctness must the algorithms satisfy?

Question 2: How to define a security model?

Question 2: How to define a security model?

A security model can be defined via a game played between adversary and challenger.

- The challenger is the secret key owner of a cryptosystem.
- The adversary is trying to break the cryptosystem.

The challenger must know something that the adversary doesn't know.

A security model defines:

- The adversary's capabilities:
  - What information the adversary can query
  - When the adversary can query information
- The adversary's security goal:
  - How the adversary wins the game (breaks the scheme).

Question 3: What should a definition for security model satisfy?

Question 3: What should a definition for security model satisfy?

When defining a security model, we don't consider

- The adversary's trivial attack that can help the adversary break the cryptosystem easily. For example, the adversary asks the challenger to share all secrets (e.g. all secret keys) with it.
- The adversary's strategy about how it obtains a piece of information. Therefore, when the adversary makes a query, the challenger must respond to this query correctly and honestly.

When defining a security model, make sure that

the advantage is negligible.

Question 4: How many security models do we need in definition?

Question 4: How many security models do we need in definition?

- A cryptosystem might have more than one security service.
- Each security service could need one security model.

  For example:

  signcryption = signature + encryption

  Two security models:
  cannot break signature, cannot break encryption
- A security model could capture multiple security definitions (services). An example is anonymous receiver with message indistinguishability in IBE.

Question 5: Why prime field is important?

Question 5: Why prime field is important?

$(\mathbb{F}_q, +, *)$

- There are q elements in this field $\mathbb{Z}_q = \{0, 1, 2, \cdots, q-1\}$.
- $u + v = u+v \mod q$.
- $u * v = u*v \mod q$.
- $-u = q-u$.
- $u^{-1} = u^{q-2} \mod q$.

Note: Prime field is important due

- Computations over group and pairing have this operation.
- $g^{\frac{1}{x}}$ is computable for any $x \in \mathbb{Z}_p^*$.

Question 6: How to quickly compute an exponentiation?

Question 6: How to quickly compute an exponentiation?

Given $g \in \mathbb{G}$ and $x \in \mathbb{Z}_p$, we can compute g^x as follows.

- Convert x into an n-bit string x:

$$x = x_{n-1} \cdots x_1 x_0 = \sum_{i=0}^{n-1} x_i 2^i.$$

- Let $g_i = g^{2^i}$. Compute $g_i = g_{i-1} \cdot g_{i-1}$ for all $i \in [1, n-1]$.
- Compute g^x by

$$g^x = \prod_{i=0}^{n-1} g_i^{x_i} = g^{\sum_{i=0}^{n-1} x_i 2^i}.$$

Question 7: What should we consider when choosing a group?

Question 7: What should we consider when choosing a group?

A group $(\mathbb{G}, g, p)$ whose DL problem is hard. We must consider generic attack and specific attack in solving the DL problem.

- The parameter must satisfy $p \geq 2^{160}$ to resist generic attacks. (The algorithm that can solve DLP defined over any group is associated with p.)
- All other parameters for specific group constructions, such as the size of group element, must be large enough to resist specific attacks for solving DLP.

Question 8: What are differences between Modular Multiplicative Group and Elliptic Curve Group?

Question 8: What are differences between Modular Multiplicative Group and Elliptic Curve Group?

$(\mathbb{G}, g, q, p)$.

- **Group Elements.** The group elements are integers from
$$\mathbb{Z}_q^* = \{1, 2, \cdots, q-1\}$$
- **Group Generator.** g is from $\mathbb{Z}_q^*$.
- **Group Order.** $p|(q-1)$.
- **Group Operation.** We have $u \cdot v = u \times v \bmod q$.

Note: $|g| \geq 1024$ for 80-bit security.

$(\mathbb{G}, g, p)$.

- **Group Elements.** The group elements are points on the elliptic curve.
- **Group Generator.** g is a point.
- **Group Order.** p a prime order.
- **Group Operation.** defined by elliptic curves.

Note: $|g| \geq 160$ for 80-bit security.

Question 9: What kinds of pairing can we construct and what are their representation size?

Question 9: What kinds of pairing can we construct and what are their representation size?

Bilinear pairing ($\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$) falls into the following three types.

- Symmetric. $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. Denoted by $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

$$|g| \geq 512(\text{bits}), \quad |e(g,g)| \geq 1024(\text{bits}).$$

- Asymmetric 1. $\mathbb{G}_1 \neq \mathbb{G}_2$ with homomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$.

$$|g_1| \geq 160(\text{bits}), |g_2| \geq 1024(\text{bits}), \quad |e(g,g)| \geq 1024(\text{bits}).$$

- Asymmetric 2. $\mathbb{G}_1 \neq \mathbb{G}_2$ with no efficient homomorphism.

$$|g_1| \geq 160(\text{bits}), |g_2| \geq 512(\text{bits}), \quad |e(g,g)| \geq 1024(\text{bits}).$$

Question 10: Do you know the applications of hash functions based on their output spaces?

Question 10: Do you know the applications of hash functions based on their output spaces?

- $H : \{0,1\}^* \to \{0,1\}^n$. The output space is the set containing all n-bit strings. We mainly use this kind of hash function to generate a symmetric key from the key space $\{0,1\}^n$ for hybrid encryption.
- $H : \{0,1\}^* \to \mathbb{Z}_p$. The output space is $\{0,1,2,\cdots,p-1\}$, where p is the group order. We use this kind of hash function to embed hashing values in group exponents such as $g^{H(m)}$.
- $H : \{0,1\}^* \to \mathbb{G}$. The output space is a cyclic group. That is, this hash function will hash the input string into a group element. This hash function exists for some groups only.

Question 11: What are differences between computational problem and decisional problem?

Question 11: What are differences between computational problem and decisional problem?

In a computational problem,

- The solution y to the problem instance x is from a large space. For example, the space size is up to $2^{|x|}$.
- CDH problem: instance $x = (g, g^a, g^b) \in \mathbb{G}$; solution $y = g^{ab} \in \mathbb{G}$.

In a decisional problem,

- The solution y to the problem instance x is from the set $\{0, 1\}$ with only two answers. If $y = 1$, the instance x is also called true instance. Otherwise, $y = 0$ and x is called false instance.
- DDH problem: instance $x = (g, g^a, g^b, Z) \in \mathbb{G}$; solution $y \in \{0, 1\}$

$$y = \begin{cases} 1 & Z = g^{ab} \\ 0 & Z \neq g^{ab} \end{cases}$$

Question 12: What are differences between deterministic and probabilistic (algorithm)?

Question 12: What are differences between deterministic and probabilistic (algorithm)?

- A deterministic algorithm is an algorithm where, given as an input a problem instance x, it will always return a correct result (solution y).
- A probabilistic (randomized) algorithm is an algorithm where given as an input a problem instance, it will return a correct result with some likelihood.
- Probabilistic algorithms are believed to be more efficient than deterministic algorithms in solving problems

We denote by (t, ϵ) that an algorithm returns a correct result in time t with success probability ϵ.

Question 13: What are polynomial time and exponential time?

Question 13: What are polynomial time and exponential time?

Let $f(\lambda)$ be the time cost function of solving a computing problem whose instance length is λ.

- Polynomial time means that there exists $c, k > 0$ such that

$$f(\lambda) \leq c \cdot \lambda^k.$$

- Exponential time means that there exists $c > 0$ such that

$$2^{c \cdot \lambda} \leq f(\lambda)$$

Note: Computational complexity has different definitions on polynomial and exponential but the above is what we want in cryptography. Easy should be easy. Hard cannot become easy.

Question 14: What is the motivation of defining "advantage" instead of "probability"?

Question 14: What is the motivation of defining "advantage" instead of "probability"?

- The concept "advantage" was motivated by the inconsistent probabilities in different events. The probability cannot be set as a value to distinguish successful attack from failed attack.
- Advantage is an adjusted probability to replaced probability. The minimal advantage in definition must be zero.

Let P_{ideal} be the maximal probability of breaking a secure scheme.

- If P_{ideal} is non-negligible, we define the advantage as

$$\text{Advantage} = \text{Probability of Successful Attack} - P_{ideal}.$$

- If P_{ideal} is negligible, we define the advantage as

$$\text{Advantage} = \text{Probability of Successful Attack}.$$

Question 15: How to prove that a computing problem is hard?

Question 15: How to prove that a computing problem is hard?

- Reduction. We prove that solving problem A implies solving an existing problem B that is well believed to be hard. (Will be introduced more in next lecture).
- Weak Computational Model. We prove that solving problem A in a weak computational model is hard. One of such a model is the generic group model for proving hard problems over a cyclic group.
- Membership proof.
  - A family of computing problems are “proved” to be hard.
  - We prove that problem A belongs to this family.

Question 16: What are differences between security parameter and security level?

Question 16: What are differences between security parameter and security level?

- Taking as input a security parameter λ, the key generation algorithms outputs a public key and a secret key.
- The security parameter is the bit length of input string.
- security parameter can be roughly seen as instance length.
- In group-based cryptography, the security parameter λ refers to the bit length of a group element, such as 160 bits or 1,024 bits.
- Breaking a scheme is harder when the length of λ grows.
- The security level indicates the strength (time cost) of an adversary in breaking a scheme.
- When λ is security parameter, we have $f(\lambda)$ is called security level.

Question 17: What should we consider when defining security of a scheme?

Question 17: What should we consider when defining security of a scheme?

Definition (Security)

We say that a scheme generated with security parameter λ is secure in a security model if every PPT adversary has advantage $\epsilon(\lambda)$ only, where $\epsilon(\lambda)$ is a negligible function in λ.

Note:

- The advantage must be well defined in the security model.
- Security definition must clearly define its security model.

Outline

Question 18: What is reduction and how to program a reduction in computational complexity?

Question 18: What is reduction and how to program a reduction in computational complexity?

The problem A is reducible to the problem B

- 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 back to problem solution y_A.
- Problem A is not harder than problem B.

Question 19: What are the basic components in security reduction?

Question 19: What are the basic components in security reduction?

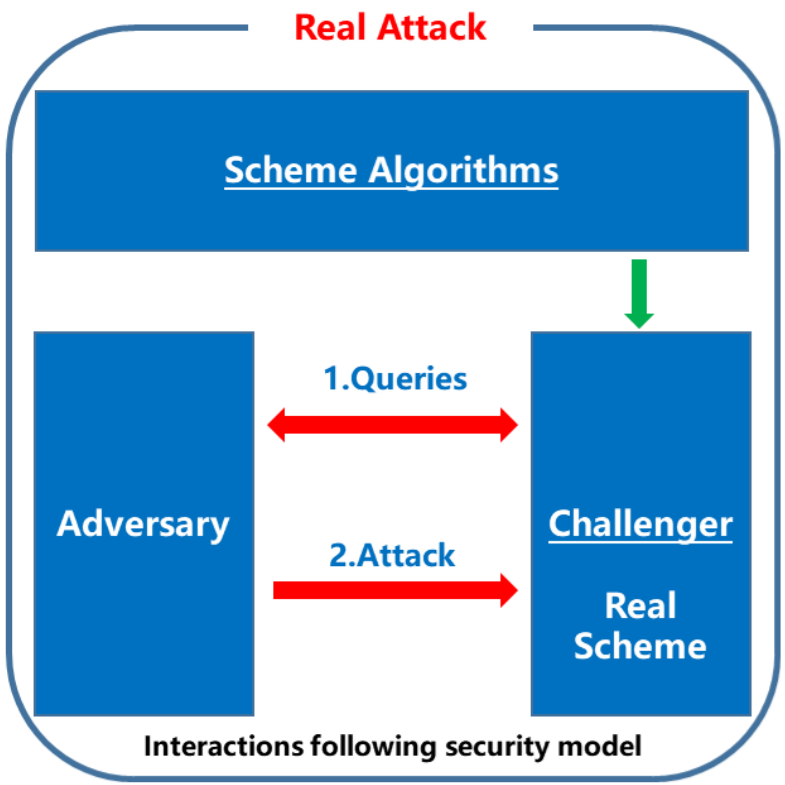
Proof. Suppose there exists an adversary who can break the proposed scheme (in a security model).

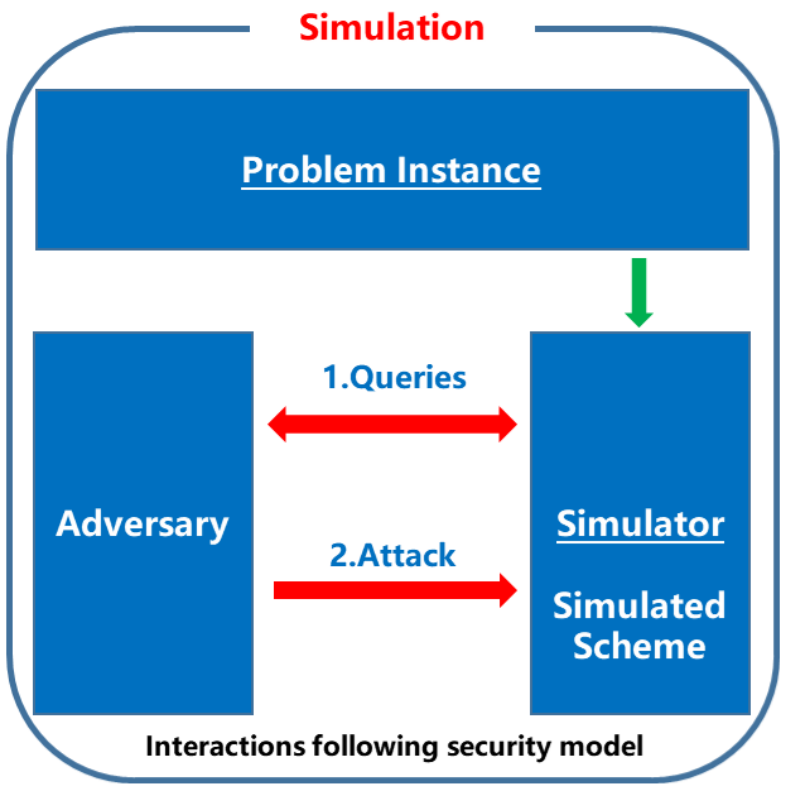
- **Simulation.** generates a simulated scheme (using problem instance x_P) and interacts with the adversary by following the security model.
- **Solution.** extracts problem solution y_P with the help of the adversary's attack on the simulated scheme.
- **Analysis.** show that the advantage of solving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

□

Question 20: What are differences between real attack and simulation?

Question 20: What are differences between real attack and simulation?





All differences are given in the above table.

Question 21: Why tight reduction is theoretically important in security reduction?

Question 21: Why tight reduction is theoretically important in security reduction?

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

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

We have

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

We obtain

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

- The security level of scheme S is low if L and T are high.
- We don't know the exact security level of scheme S except that it is higher than the lower bound security level.

Question 22: What is ideal security reduction?

Question 22: What is 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.

Question 23: Why do we need an indistinguishable simulation?

Question 23: Why do we need an indistinguishable simulation?

- 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.

Question 24: How many types of “attacks” are defined in security reduction?

Question 24: How many types of "attacks" are defined in security reduction?

- **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.
- **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.

Question 25: What is the main feature of black-box adversary?

Question 25: What is the main feature of 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 adaptively chosen, we have

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

Note: **Adaptive = Unpredictable.** The adaptive output from the black-box adversary includes

- Adaptive query.
- Adaptive attack.

Question 26: Why do we amplify black-box adversary into malicious adversary?

Question 26: Why do we amplify black-box adversary into malicious adversary?

- The probability of a potential output 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.
- The probability of returning a useful attack by the adversary must be larger than that by a malicious adversary.

Question 27: How to understand computationally unbounded adversary in security reduction?

Question 27: How to understand computationally unbounded adversary in security reduction?

- 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.
- If a security reduction works against a computationally unbounded adversary, it must work against a computationally bounded adversary.

Question 28: What does the adversary know?

Question 28: What does the adversary know?

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.

Question 29: What does the adversary never know?

Question 29: What does the adversary never know?

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.)

Question 30: What do we concern when the adversary attacks the simulated scheme?

Question 30: What do we concern when the adversary attacks the simulated scheme?

- 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.
- The adversary can always launch a useless attack.

Question 31: When is a security reduction correct?

Question 31: When is a security reduction correct?

- 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).

Question 32: What are paradoxes when the secret key is unknown in simulation?

Question 32: What are paradoxes when the secret key is unknown in simulation?

- In simulated scheme, the simulator can compute signatures on messages queried by the adversary without knowing the secret key.
- In real scheme, signatures must be computed with secret key. Otherwise, the signature scheme must be insecure!

Why the signature computing in simulated scheme is so special?

- In simulated scheme, the simulator uses the signature generated by the adversary to solve hard problem.
- In simulated scheme, the simulator can also generate some signatures by itself but the simulator cannot use these signatures to solve hard problem.

Why the forged signature from the adversary is so special?

Question 33: What are paradoxes when the secret key is known in simulation?

Question 33: What are paradoxes when the secret key is known in simulation?

Paradox in Signature

- In simulated scheme, the simulator uses the signature generated by the adversary to solve hard problem.
- In simulated scheme, the simulator knows the secret key and can compute signatures on all messages.

Why the forged signature from the adversary is so special?

Paradox in Encryption

- In simulated scheme, the simulator uses the adversary's decryption result to solve hard problem.
- In simulated scheme, the simulator knows the secret key and can decrypt all ciphertexts including the challenge ciphertext.

Why the decryption result from the adversary is so special?

Question 34: What is trapdoor in security reduction?

Question 34: What is trapdoor in security reduction?

- Trapdoor is the additional secret in simulated scheme for computing response when the secret key is unknown.
- We program the simulation in the way that computing using secret key can be replaced with computing using the trapdoor.
- The trapdoor is also used to extract problem solution from the adversary's attack on the simulated scheme.

Question 35: What should we do when proving security in random oracle model?

Question 35: What should we do when proving security in random oracle model?

- Proving security of a proposed scheme in the random oracle model requires at least one hash function to be set as a random oracle. However, it doesn't need all hash functions to be set as random oracles (it depends on security reduction).
- When a hash function is set as a random oracle, the adversary will not receive the hash function algorithm from the simulator in the security proof. The adversary can only access the "hash function" by asking the random oracle.
- A random oracle is an ideal hash function. However, we don't need to construct such an ideal hash function in the simulation. Instead, the main task in the simulator is to think how to respond to each input query.

Question 36: How to apply random oracle in security reduction?

Question 36: How to apply random oracle in security reduction?

Input	Random Oracle	Output
x_1		y_1
x_2		y_2
x_3	Simulator	y_3
$\vdots$		$\vdots$
x_q		y_q

There are three different ways in the application. They are

- We program the output and know a trapdoor.
- We program the output into a special space.
- We control the output and know the input.

Question 37: How to prove random and independent?

Question 37: How to prove random and independent?

Lemma

Suppose a real scheme and a simulated scheme generate integers (A, B, C) with different methods described as follows.

- *In the real scheme, (A, B, C) are randomly chosen from $\mathbb{Z}_p$.*
- *In the simulated scheme, (A, B, C) are computed by a function with random (w, x, y, z) from $\mathbb{Z}_p$ denoted by $(A, B, C) = F(w, x, y, z)$.*

Suppose the adversary knows the function F from the reduction algorithm but not (w, x, y, z). The simulated scheme is indistinguishable from the real scheme if for any given (A, B, C) from $\mathbb{Z}_p$, the number of solutions (w, x, y, z) satisfying $(A, B, C) = F(w, x, y, z)$ is the same.

Question 38: How to prove the adversary has no advantage in launching useless attack?

Question 38: How to prove the adversary has no advantage in launching useless attack?

Let $Attack_i$ be one specific way for breaking the proposed scheme. That is, launching a specific way of queries and challenge.

- Let $\{Attack_1, Attack_2, \cdots, Attack_n\}$ be the set of all potential attacks.
- Some attacks are useful attacks and some are useless attacks.
- The adversary will launch an adaptive $Attack^*$ from the set.

The computationally unbounded adversary has no advantage in identifying which attack is a useless attack. Otherwise, the adaptive attack will be useless.

Question 39: What should we analyze in security reduction?

Question 39: What should we analyze in security reduction?

Proof. Suppose there exists an adversary.....

Simulation + Solution

This completes..... The correctness is analyzed as follows.

Indistinguishable simulation. Analyze that the simulation is indistinguishable when simulation is successful.

Probability of successful simulation and useful attack. Analyze the success probability of solving hard problem.

Advantage and time cost. Analyze advantage and time cost of solving hard problem.

This completes the proof. □

Outline

Question 40: What is the proof structure for digital signatures?

Question 40: What is the proof structure for digital signatures?

A security proof is composed of the following three parts.

- **Simulation.** The simulator uses the problem instance to generate a simulated scheme and interacts with the adversary following the unforgeability security model.
- **Solution.** The simulator solves the underlying hard problem using the forged signature generated by the adversary.
- **Analysis.** In this part, we need to provide the following analysis.
  1. The simulation is indistinguishable from the real attack.
  2. The probability P_S of successful simulation.
  3. The probability P_U of useful attack.
  4. The advantage ϵ_R of solving the underlying hard problem.
  5. The time cost of solving the underlying hard problem.

Note: Most security reductions use the forged signature to solve hard problem but it is not the only choice.

Question 41: What are simulatable and reducible?

Question 41: What are simulatable and reducible?

If problem solution is extracted from the adversary's forged signature, we can classify all signatures into two types: *simulatable* and *reducible*.

- **Simulatable.** A signature is simulatable if it can be computed by the simulator.
- **Reducible.** A signature is reducible if it can be used to solve the underlying hard problem.

Note: If a signature on m generated with random number r is simulatable, it doesn't mean that the signature on m generated with another random number r' must be also simulatable.

Question 42: How to achieve tight reductions for digital signatures?

Question 42: How to achieve tight reductions for digital signatures?

Each signature in simulation should be either simulatable or reducible.

A successful security reduction requires that

- All queried signatures are simulatable;
- The forged signature is reducible.

A security reduction is tight in the standard security model if

- No matter what the queried messages $(m_1, m_2, m_3, \cdots, m_q)$ are, their signatures are simulatable.
- No matter what the forged signature (m^*, σ_{m^*}) is, the forged signature is reducible.

Question 43: How to setup partition in security reduction for digital signatures?

Question 43: How to setup partition in security reduction for digital signatures?

There are two ways.

Normal. A reduction algorithm provides one simulation.

- There is one partition.
- The adversary has no advantage in computing the partition.

Advanced. A reduction algorithm provides two simulations.

- There are two partitions (one in each simulation).
- The adversary can know the partitions.
- The simulator will randomly choose one in simulation.
- The adversary cannot distinguish which one is chosen.
- The adversary cannot find a message whose signature is simulatable in both two simulations.

Question 44: What is the proof structure for encryption under decisional assumption?

Question 44: What is the proof structure for encryption under decisional assumption?

- **Simulation.** The simulator uses the problem instance (X, Z) to generate a simulated scheme.
- **Solution.** The simulator solves the decisional hard problem using the adversary's guess c' of c, where the message in the challenge ciphertext is m_c. Guessing Z is the same in all security reductions.
  - If $c' = c$, the simulator outputs that Z is true.
  - Otherwise, $c' \neq c$, and it outputs that Z is false.
- **Analysis.** In this part, we need to provide the following analysis.
  1. The simulation is indistinguishable from the real attack if Z is true.
  2. The probability P_S of successful simulation.
  3. The probability P_T of breaking the challenge ciphertext if Z is true.
  4. The probability P_F of breaking the challenge ciphertext if Z is false.
  5. The advantage ϵ_R of solving the underlying hard problem.
  6. The time cost of solving the underlying hard problem.

Question 45: What should the simulation satisfy when Z is true and false?

Question 45: What should the simulation satisfy when Z is true and false?

$$\epsilon_R = P_S(P_T - P_F) = P_S\left(\frac{1}{2} + \frac{\epsilon}{2} - P_F\right).$$

The simulation should satisfy that

- The probability of successful simulation P_S is non-negligible.
- The simulated scheme is indistinguishable if Z is true.
- $P_F \approx \frac{1}{2}$ if Z is false. It means that the malicious adversary has almost no advantage in breaking the false challenge ciphertext.

Question 46: What should the decryption simulation satisfy for CCA security?

Question 46: What should the decryption simulation satisfy for CCA security?

$$\epsilon_R = P_S(P_T - P_F) = P_S\left(\frac{1}{2} + \frac{\epsilon}{2} - P_F\right).$$

If Z is true, the simulation of decryption requires that

- The adversary will still break the challenge ciphertext with $\frac{1}{2} + \frac{\epsilon}{2}$.
- The decryption simulation should be indistinguishable.
- We don't care the decryption simulation helps the adversary.

If Z is false, the simulation of decryption requires that

- P_F is still as small as $\frac{1}{2}$.
- We do care the decryption simulation helps the adversary.
- We don't care distinguishable decryption simulation due to false Z.

Question 47: What is the technical idea of security reduction for encryption under computational hardness assumption?

Question 47: What is the technical idea of security reduction for encryption under computational hardness assumption?

Let $H : \{0,1\}^* \to \{0,1\}^n$ be a random oracle. Let X be a given problem instance, and y be its solution. Suppose the challenge ciphertext is

$$CT = \Big(X,\ H(y) \oplus m_c\Big).$$

- Suppose the adversary can break the challenge ciphertext.
- The adversary must query y to the random oracle. Otherwise, without querying y to the random oracle, CT is equivalent to a one-time pad, which is contrary to the breaking assumption.
- We program the reduction in a way that y is the problem solution.

Question 48: What is the proof structure for encryption under computational hardness assumption?

Question 48: What is the proof structure for encryption under computational hardness assumption?

- **Simulation.** The simulator programs the random oracle simulation, generates the simulated scheme using the received problem instance, and interacts with the adversary.
- **Solution.** The simulator solves the computational hard problem with hash queries to the random oracle.
- **Analysis.** In this part, we need to provide the following analysis.
  1. The simulation is indistinguishable from the real attack if no challenge hash query is made by the adversary.
  2. The probability P_S of successful simulation.
  3. The adversary has no advantage in breaking the challenge ciphertext if it doesn't make the challenge hash query to the random oracle.
  4. The probability P_C of finding the correct solution from hash queries.
  5. The advantage ϵ_R of solving the underlying hard problem.
  6. The time cost of solving the underlying hard problem.

Question 49: Is the simulation wrong if the adversary can distinguish the simulation?

Question 49: Is the simulation wrong if the adversary can distinguish the simulation?

- Before the adversary makes a challenge hash query, we require indistinguishable simulation.
- After the adversary makes a challenge hash query, we don't care any simulation result!

Note: The above is given for encryption under computational hardness assumption only.

Question 50: How to simulate decryption for encryption under computational hardness assumption?

Question 50: How to simulate decryption for encryption under computational hardness assumption?

- Firstly, we can use hash queries instead of the challenge decryption key to simulate the decryption. A ciphertext is valid if and only if the adversary ever made the correct hash query to the random oracle. This condition is necessary if the simulator is to simulate the decryption correctly. (Inputs to random oracle are set as trapdoor)
- Secondly, there should be a mechanism for checking which hash query is the correct hash query for a decryption. Otherwise, given a ciphertext for a decryption query, the simulator might return many distinct results depending on the used hash queries.

Outline

This is the so called Security Reduction

My IQ is up to 186.

My interest is breaking schemes.

~~*You want me to help you solve problem? Fool me first!*~~

I can help you solve problem, as long as you can fool me!

Goooooooooooooooood Luck!


END


Introduction
to Security
Reduction