Introduction to Security Reduction

Lecture 10: Security Proofs

(Encryption under Computational Hardness Assumption)

Adversary

My IQ is up to 186.

My interest is breaking schemes.

You want me to help you solve problem?

Fool me first!

Computational Complexity Theory

Outline

1 Random Oracle in Applications

- Random and Independent Revisited
- One-Time Pad Revisited
- Solution to Hard Problem Revisited
- Simulation of Challenge Ciphertext

2 Proof Structure

- Proof Structure
- Advantage Analysis

3 CCA Security

4 Summary

Outline

Outline

Random and Independent Revisited

Let H be a cryptographic hash function, and x be a random input string.

- If H is a cryptographic hash function, $H(x)$ is dependent on x and the hash function algorithm. That is, $H(x)$ is computable from x and the hash function H.
- If H is set as a random oracle, $H(x)$ is random and independent of x. This is due to the fact that $H(x)$ is an element randomly chosen by the simulator.

In a security proof with random oracles, if the adversary doesn't query x to the random oracle, $H(x)$ is random and unknown to the adversary.

This is the core of security reduction for encryption with random oracles.

Outline

One-Time Pad Revisited

Let $H : \{0,1\}^* \rightarrow \{0,1\}^n$ be a cryptographic hash function. We consider the encryption of $m_c \in \{m_0, m_1\}$ with an arbitrary string x.

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

- If H is a hash function, the above ciphertext is not a one-time pad. Given x and the hash function H, the adversary can compute $H(x)$ by itself and decrypt the ciphertext to obtain the message m_c.
- If H is set as a random oracle, the adversary cannot compute $H(x)$ by itself and must query x to the random oracle to know $H(x)$.

Introduction to Security Reduction

One-Time Pad Revisited

Let $H : \{0,1\}^* \rightarrow \{0,1\}^n$ be a cryptographic hash function. We consider the encryption of $m_c \in \{m_0, m_1\}$ with an arbitrary string x.

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

- If H is a hash function, the above ciphertext is not a one-time pad. Given x and the hash function H, the adversary can compute $H(x)$ by itself and decrypt the ciphertext to obtain the message m_c.
- If H is set as a random oracle, the adversary cannot compute $H(x)$ by itself and must query x to the random oracle to know $H(x)$.
  - **Before Querying x.** Since $H(x)$ is random and unknown to the adversary, the adversary has no advantage in guessing the message m_c except with probability $\frac{1}{2}$.
  - **After Querying x.** Once the adversary queries x to the random oracle and receives the response $H(x)$, the adversary can guess the encrypted message with advantage 1.

Outline

Solution to Hard Problem Revisited

Let $H : \{0,1\}^* \rightarrow \{0,1\}^n$ 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.

Therefore, the solution to the problem instance will appear in one of the hash queries made by the adversary.

Introduction to Security Reduction

Outline

Simulation of Challenge Ciphertext

Background: Suppose the challenge ciphertext in the real scheme is

$$CT^* = \Big(g, g^a, g^b, H(g^{ab}) \oplus m_c\Big).$$

In the simulated scheme, the simulator cannot directly simulate $H(g^{ab}) \oplus m_c$ because g^{ab} is unknown to the simulator(CDH problem).

Simulation of Challenge Ciphertext

Background: Suppose the challenge ciphertext in the real scheme is

$$CT^* = \Big(g, g^a, g^b, H(g^{ab}) \oplus m_c\Big).$$

In the simulated scheme, the simulator cannot directly simulate $H(g^{ab}) \oplus m_c$ because g^{ab} is unknown to the simulator(CDH problem).

Solution: Fortunately, this problem can be easily solved.

To be precise, the simulator chooses a random string $R \in \{0,1\}^n$ to replace $H(y) \oplus m_c$. The challenge ciphertext in the simulated scheme is

$$CT^* = (g, g^a, g^b, R).$$

The challenge ciphertext can be seen as an encryption of the message $m_c \in \{m_0, m_1\}$ if $H(y) = R \oplus m_c$, where we have

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

Simulation of Challenge Ciphertext

$$CT^* = (g, g^a, g^b, R).$$

- Unfortunately, after sending this challenge ciphertext to the adversary, the simulator may not know which hash query from the adversary is the solution y and will respond to the query on y with a random element different from $R \oplus m_c$.
- Therefore, the query response is wrong, and the challenge ciphertext in the simulation is distinguishable from that in the real scheme. The adversary finds that the challenge ciphertext is an encryption of message not in $\{m_0, m_1\}$.

Simulation of Challenge Ciphertext

$$CT^* = (g, g^a, g^b, R).$$

- **Before Querying g^{ab}.** From the point of view of the adversary, the challenge ciphertext is an encryption of m_0 if $H(g^{ab}) = R \oplus m_0$ and an encryption of m_1 if $H(g^{ab}) = R \oplus m_1$. Without making a hash query on y to the random oracle, the adversary never knows $H(g^{ab})$, and thus has no advantage in breaking CT^*.
- **After Querying g^{ab}.** Once the adversary makes a hash query on g^{ab} to the random oracle, the simulation is distinguishable. However, we don't care because the simulator has already received the g^{ab} from the adversary and can solve the underlying hard problem.

Definition. We denote by Q^* the *challenge hash query*, if the adversary has no advantage in guessing the encrypted message without making a hash query on Q^* to the random oracle.

A Small Summary

- Indistinguishable simulation is desired in security reduction.
- Without this feature, we cannot use the adversary's attack ability.
- However, we only require this property before the adversary launches a useful attack.

Note: The encryption under decisional hardness assumption requires IND if Z is true. It is different from the required

Outline

Outline

Proof Structure

- **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.

Outline

Lemma

Lemma
If the adversary has no advantage in breaking the challenge ciphertext without querying Q^* to the random oracle, the adversary will make the challenge hash query to the random oracle with probability ϵ.

Proof. According to the breaking assumption, we have

$$\Pr[c' = c] = \frac{1}{2} + \frac{\epsilon}{2}.$$

Let H^* denote the event of making the challenge hash query to the random oracle, and H^{*c} be the complementary event of H^*.

$$\Pr[c' = c | H^*] = 1, \quad \Pr[c' = c | H^{*c}] = \frac{1}{2}.$$

$$\begin{aligned} \Pr[c' = c] &= \Pr[c' = c|H^*] \Pr[H^*] + \Pr[c' = c|H^{*c}] \Pr[H^{*c}] \\ &= \Pr[H^*] + \frac{1}{2}(1 - \Pr[H^*]) = \frac{1}{2} + \frac{1}{2}\Pr[H^*], \end{aligned}$$

and deduct $\Pr[H^*] = \epsilon$. This completes the proof. □

Advantage

The advantage ϵ_R of solving the underlying computational hard problem is defined as

$$\epsilon_R = P_S \cdot \epsilon \cdot P_C.$$

Suppose that

- the simulation is successful and indistinguishable, and
- the adversary has no advantage in breaking the challenge ciphertext without making the query on Q^*.

The challenge hash query will appear in the hash list with probability ϵ.

Finally, the probability of picking Q^* from the hash list is P_C.

Therefore, the advantage of solving hard problem is $P_S \cdot \epsilon \cdot P_C$.

The Probability P_C

The simulator needs to pick one hash query as the challenge hash query and extracts the solution to the problem instance from it.

- The decisional variant of the computational hard problem is hard. The simulator cannot verify which hash query is the correct one. It has to randomly pick one of the hash queries as the challenge hash query. We have $P_C = 1/q_H$.
- The decisional variant of the computational hard problem is easy. The simulator can verify which hash query is the challenge hash query. The reason is that the simulator can test all the hash queries one by one until it finds the challenge hash query. We have $P_C = 1$.

Outline

Simulation of Decryption

In a security reduction for an encryption scheme under a decisional hardness assumption,

- the decryption simulation must be indistinguishable from the real attack if Z is true, and
- the decryption simulation must not help the adversary break the false challenge ciphertext if Z is false.

However, the requirements of the decryption simulation for this type of security reduction are slightly different because there is no Z.

Simulation of Decryption

Requirements on Decryption 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: what do we need to achieve indistinguishable decryption simulation?

Simulation of Decryption

Requirements on Indistinguishable Decryption Simulation:

- If a ciphertext is generated and is independent of the challenge ciphertext, it will be accepted or rejected the same as the real scheme.
- If a ciphertext is generated or modified from the challenge ciphertext, it will be rejected the same as the real scheme.

Note: The challenge ciphertext in the simulated scheme is a fake ciphertext (X, R) while in the real scheme is a real ciphertext $(X, H(y) \oplus m_c)$. They are different and cannot be distinguished. We simply require the scheme to reject any ciphertext modified from the challenge ciphertext.

Fuchun Guo · Willy Susilo · Yi Mu
Introduction to Security Reduction
Springer

Decryption Simulation Without Key

- In this type of security reduction, the challenge decryption key is usually programmed as a key unknown to the simulator.
- We use the help of random oracle in the decryption simulation. Namely, any input to hash must be queried.

The key pair is $pk = (g, g_1, H)$ and $sk = \alpha$, where $g_1 = g^\alpha$.

The encryption algorithm chooses a random $r \in \mathbb{Z}_p$ and computes:

$$CT = (C_1, C_2, C_3) = \Big(g^r,\ H(0||g_1^r) \oplus r,\ H(1||g_1^r) \oplus m\Big).$$

This encryption scheme is not IND-CCA secure but IND-CCA1 secure.

Question: How to simulate decryption when α is unknown.

Decryption Simulation Without Key

Answer is given in the next page.

Decryption Simulation Without Key

$$CT = (C_1, C_2, C_3) = \Big(g^r,\ H(0||g_1^r) \oplus r,\ H(1||g_1^r) \oplus m \Big).$$

A queried ciphertext CT is valid if and only if CT can be re-created with the decrypted random number r' and the decrypted message m'.

That is,

$$\Big(g^{r'},\ H(0||g_1^{r'}) \oplus r',\ H(1||g_1^{r'}) \oplus m' \Big) = CT.$$

Observation: Without querying $0||g_1^r$ to the random oracle, the adversary cannot correctly compute C_2 even the adversary knows r.

The simulator uses the queried $0||g_1^r$ to decrypt ciphertext and verify it.

Introduction to Security Reduction

Decryption Simulation Without Key

$$CT = (C_1, C_2, C_3) = \Big(g^r,\ H(0||g_1^r) \oplus r,\ H(1||g_1^r) \oplus m\Big).$$

Suppose $(x_1, y_1), (x_2, y_2), \cdots, (x_q, y_q)$ are in the hash list, where x, y denote a query and a response, respectively. If CT is a valid ciphertext, one of the hash queries must be equal to $0||g_1^r$. Otherwise, the ciphertext is invalid.

- For all $i \in [1, q]$, it starts with $i = 1$ and computes $r' = y_1 \oplus C_2$.
- It uses r' to decrypt the message m' by computing $H(1||g_1^{r'}) \oplus C_3$.
- It checks whether the ciphertext CT can be re-created with (r', m'). If yes, the simulator returns message m'. Otherwise, the simulator sets $i = i + 1$ and repeats the above procedure.

If all y_i cannot decrypt the ciphertext correctly, the simulator outputs $\perp$ as the decryption result on the queried ciphertext CT. That is, CT is invalid.

Decryption Simulation Without Key

- 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

Summary

A correct security reduction for a public-key encryption scheme under a computational hardness assumption must satisfy the conditions.

- The underlying hard problem is a computational problem.
- The simulator doesn't know the secret key.
- The simulator can simulate the decryption for CCA security.
- The probability of successful simulation is non-negligible.
- Without making the challenge hash query Q^*, the adversary cannot distinguish the simulated scheme from the real scheme and has no advantage in breaking the challenge ciphertext.
- The simulator uses Q^* to solve the hard problem.
- The advantage ϵ_R of solving the hard problem is non-negligible.
- The time cost of the simulation is polynomial time.

Have a Try?

KeyGen: The key generation algorithm randomly chooses $\alpha \in \mathbb{Z}_p$, computes $g_1 = g^\alpha$, and returns a public/secret key pair (pk, sk) as follows:

$$pk = (g, g_1, H), \quad sk = \alpha.$$

Encrypt: The encryption algorithm takes as input a message $m \in \{0,1\}^n$, the public key pk, and the system parameters SP. It chooses a random number $r \in \mathbb{Z}_p$ and returns the ciphertext CT as

$$CT = (C_1, C_2) = \Big(g^r,\ H(g_1^r) \oplus m\Big).$$

Decrypt: The decryption algorithm takes as input a ciphertext CT, the secret key sk, and the system parameters SP. Let $CT = (C_1, C_2)$. It decrypts

$$C_2 \oplus H(C_1^\alpha) = H(g_1^r) \oplus m \oplus H\big(g^{\alpha r}\big) = m.$$

It is secure in IND-CPA security model under CDH assumption.

END

Introduction to Security Reduction