Introduction to Security Reduction

Lecture 8: Security Proofs

(Digital Signatures)

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 Proof Structure

2 Simulatable and Reducible

3 Partition
- Partition:Standard
- Partition:Advanced

4 Summary

Outline

1. **Proof Structure**
2. Simulatable and Reducible
3. Partition
   - Partition:Standard
   - Partition:Advanced
4. Summary

Proof Structure

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.

Advantage Calculation

Let ϵ be the advantage of the adversary in breaking the proposed signature scheme. The advantage of solving the underlying hard problem, denoted by ϵ_R, is

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

- The simulation is successful and indistinguishable from the real attack with probability P_S.
- Then the adversary can successfully forge a valid signature with probability ϵ.
- Then the forged signature is a useful attack with probability P_U and the forged signature can be reduced to solving the hard problem.

Therefore, we obtain ϵ_R as the advantage of solving the hard problem.

Advantage Calculation

Let ϵ be the advantage of the adversary in breaking the proposed signature scheme. The advantage of solving the underlying hard problem, denoted by ϵ_R, is

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

Note:

- Many security proofs only calculate the probability of successful simulation without calculating the probability of useful attack.
- Such an analysis is the same as ours because the probability of "successful simulation" in their definitions includes P_U.

The reason is due to the different definition of successful simulation.

Outline

1 Proof Structure

2 Simulatable and Reducible

3 Partition
- Partition:Standard
- Partition:Advanced

4 Summary

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.

In the security reduction for digital signature schemes, we have

- The forged signature is a useless attack if it is simulatable.
- The forged signature is a useful attack if it is reducible.

Simulatable and Reducible

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.

Simulatable and Reducible

- Simulatable and reducible are two important concepts for digital signatures and for private keys in identity-based encryption.
- We summarize three important structures used in the constructions of signature schemes and other cryptographic schemes in group-based cryptography.
- These three types are introduced in the random oracle model, where random oracles are used to decide whether a signature is simulatable or reducible.

H-Type: Hashing to Group

The H-type of signature structure is described as

$$\sigma_m = H(m)^a,$$

where $H : \{0,1\}^* \rightarrow \mathbb{G}$ is a cryptographic hash function. Here, $(g, g^a, g^b) \in \mathbb{G}$ is a CDH problem instance, and the aim is to compute g^{ab}.

Suppose H is set as a random oracle. For a query on m, the simulator responds with

$$H(m) = g^{xb+y},$$

where

- b is the unknown secret in the problem instance,
- $x \in \mathbb{Z}_p$ is adaptively chosen, and
- $y \in \mathbb{Z}_p$ is randomly chosen by the simulator.

$H(m)$ is random in $\mathbb{G}$ because y is randomly chosen from $\mathbb{Z}_p$.

H-Type: Hashing to Group

$$H(m) = g^{xb+y}$$

The simulatable and reducible conditions are described as follows:

$$\sigma_m = H(m)^a \text{ is } \begin{cases} \text{Simulatable}, & \text{if } x = 0 \\ \text{Reducible}, & \text{otherwise} \end{cases}.$$

- The H-type is simulatable if $x = 0$ because we have

$$\sigma_m = H(m)^a = (g^{0b+y})^a = g^{ya} = (g^a)^y.$$

- The H-type is reducible if $x \neq 0$ because we have

$$\left(\frac{\sigma_m}{(g^a)^y}\right)^{\frac{1}{x}} = \left(\frac{H(m)^a}{(g^a)^y}\right)^{\frac{1}{x}} = \left(\frac{g^{(xb+y)a}}{g^{ay}}\right)^{\frac{1}{x}} = \left(g^{x \cdot ab}\right)^{\frac{1}{x}} = g^{ab}.$$

C-Type: Commutative

The C-type of signature structure is described as

$$\sigma_m = \left(g^{ab} H(m)^r,\ g^r \right),$$

where $H : \{0,1\}^* \rightarrow \mathbb{G}$ is a cryptographic hash function and $r \in \mathbb{Z}_p$ is a random number. Here, $(g, g^a, g^b) \in \mathbb{G}$ is an instance of the CDH problem, and the aim is to compute g^{ab}.

Suppose H is set as a random oracle. The simulator responds to m with

$$H(m) = g^{xb+y}$$

- b is the unknown secret in the problem instance,
- $x \in \mathbb{Z}_p$ is adaptively chosen, and
- $y \in \mathbb{Z}_p$ is randomly chosen by the simulator.

$H(m)$ is random in $\mathbb{G}$ because y is randomly chosen from $\mathbb{Z}_p$.

C-Type: Commutative

$$H(m) = g^{xb+y}$$

The simulatable and reducible conditions are described as follows:

$$\sigma_m = \left(g^{ab}H(m)^r,\ g^r\right) \text{ is } \begin{cases} \text{Simulatable,} & \text{if } x \neq 0 \\ \text{Reducible,} & \text{otherwise} \end{cases}.$$

- The C-type is simulatable if $x \neq 0$ because we can choose a random $r' \in \mathbb{Z}_p$ and set $r = -\frac{a}{x} + r'$. Then, we have

$$\begin{aligned} g^{ab}H(m)^r &= (g^b)^{xr'} \cdot (g^a)^{-\frac{y}{x}} \cdot g^{r'y}, \\ g^r &= g^{-\frac{a}{x}+r'} = (g^a)^{-\frac{1}{x}} \cdot g^{r'}. \end{aligned}$$

- The C-type is reducible if $x = 0$ because we have

$$\frac{g^{ab}H(m)^r}{(g^r)^y} = \frac{g^{ab}(g^{0b+y})^r}{g^{ry}} = g^{ab}.$$

I-Type: Inverse of Group Exponent

The I-type of signature structure is described as

$$\sigma_m = h^{\frac{1}{a-H(m)}},$$

where $H:\{0,1\}^* \rightarrow \mathbb{Z}_p$ is a cryptographic hash function. Here, $(g, g^a, g^{a^2}, \cdots, g^{a^q}) \in \mathbb{G}$ is an instance of the q-SDH problem, and the aim is to compute a pair $(s, g^{\frac{1}{a+s}})$ for any $s \in \mathbb{Z}_p$.

Suppose H is set as a random oracle. The simulator responds to m with

$$H(m) = x \in \mathbb{Z}_p$$

where $x \in \mathbb{Z}_p$ is randomly chosen by the simulator, and thus $H(m)$ is random in $\mathbb{Z}_p$.

I-Type: Inverse of Group Exponent

$$H(m) = x \in \mathbb{Z}_p$$

In the simulated scheme, suppose the group element h is computed by

$$h = g^{(a-x_1)(a-x_2)\cdots(a-x_q)},$$

where a is unknown and all x_i are randomly chosen by the simulator.

$$\sigma_m = h^{\frac{1}{a-H(m)}} = g^{\frac{(a-x_1)(a-x_2)\cdots(a-x_q)}{a-H(m)}} \text{ is } \begin{cases} \text{Simulatable}, & \text{if } x \in \{x_1, x_2, \cdots, x_q\} \\ \text{Reducible}, & \text{otherwise} \end{cases}.$$

Note: The simulatable and reducible can be followed with the computation approaches in Lecture 6.

Outline

1 Proof Structure

2 Simulatable and Reducible

3 Partition
- Partition:Standard
- Partition:Advanced

4 Summary

Partition

- In the simulation, the simulator must hide from the adversary which signatures are simulatable and which signatures are reducible.
- We call the approach of splitting signatures into the above two sets **partition**. The partition decides what kinds of signatures are simulatable and reducible.
- If the adversary can always return a simulatable signature as the forged signature, the reduction will not be successful.
- The simulator must stop the adversary (who knows the reduction algorithm and can make signature queries) from finding the partition.

Note: The simulation (or reduction algorithm) decides the partition.

Partition: Two Approaches

We can program security reduction with two different approaches:

- Standard/Normal (equipped with one partition)
- Advanced/Dual (equipped with two partitions)

Note: Most security reductions used the standard approach, while the advanced approach can bring some benefits such as tight reductions.

Outline

1 Proof Structure

2 Simulatable and Reducible

3 Partition
- Partition:Standard
- Partition:Advanced

4 Summary

Partition: Standard

Normal. A reduction algorithm provides one simulation.

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

Bad Partition: Example (1)

KeyGen: $pk = (g, g_0, g_1) = (g, g^{\alpha_0}, g^{\alpha_1}), sk = (\alpha_0, \alpha_1)$.

Sign: It chooses a random $c \in \{0, 1\}$ and computes σ_m on $m \in \mathbb{Z}_p$

$$\sigma_m = g^{\frac{1}{\alpha_c + m}}.$$

Verify: It is valid if $e(\sigma_m, g_0 g^m) = e(g, g)$ or $e(\sigma_m, g_1 g^m) = e(g, g)$.

Incorrect Proof. Given as input (g, g^a), $\mathcal{B}$ runs $\mathcal{A}$ and works as follows.

Setup. The simulator randomly chooses $x \in \mathbb{Z}_p, b \in \{0, 1\}$ and sets

$$(g^{\alpha_0}, g^{\alpha_1}) = \begin{cases} (g^x,\ g^a), & \text{if } b = 0 \\ (g^a,\ g^x), & \text{otherwise} \end{cases}.$$

Query. For a signature query on m, the simulator uses α_b and computes

$$\sigma_m = g^{\frac{1}{x+m}} = g^{\frac{1}{\alpha_b + m}}.$$

Bad Partition: Example (1)

Answer is given in the next page.

Bad Partition: Example (1)

KeyGen: $pk = (g, g_0, g_1) = (g, g^{\alpha_0}, g^{\alpha_1}), sk = (\alpha_0, \alpha_1)$.

Sign: It chooses a random $c \in \{0, 1\}$ and computes σ_m on $m \in \mathbb{Z}_p$

$$\sigma_m = g^{\frac{1}{\alpha_c + m}}.$$

Incorrect Proof. The simulator randomly chooses $x \in \mathbb{Z}_p, b \in \{0, 1\}$,

$$(g^{\alpha_0}, g^{\alpha_1}) = \begin{cases} (g^x,\ g^a), & \text{if } b = 0 \\ (g^a,\ g^x), & \text{otherwise} \end{cases}.$$

Query. The simulator uses α_b and computes

$$\sigma_m = g^{\frac{1}{x+m}} = g^{\frac{1}{\alpha_b + m}}.$$

Partition: Any signature generated with the same α_b in queried signature must be simulatable.

Bad Partition: Example (2)

One-time signature where the adversary can query one signature only.

KeyGen: $pk = (g, g_1, g_2, g_3) = (g, g^\alpha, g^\beta, g^\gamma), sk = (\alpha, \beta, \gamma)$.

Sign: It chooses a random $r \in \mathbb{Z}_p$ and computes σ_m on $m \in \mathbb{Z}_p$

$$\sigma_m = \Big(r, \alpha + m\beta + r\gamma\Big).$$

Incorrect Proof. Given as input (g, g^a), $\mathcal{B}$ runs $\mathcal{A}$ and works as follows.

Setup. The simulator randomly chooses $x_1, y_1, x_2 \in \mathbb{Z}_p$ and sets

$$(\alpha, \beta, \gamma) = (a, x_1 a + y_1, a + x_2)$$

Query. For a signature query on m, the simulator uses $r = -a - x_1 m$ in simulating the signature on m.

Bad Partition: Example (2)

Answer is given in the next page.

Bad Partition: Example (2)

One-time signature where the adversary can query one signature only.

KeyGen: $pk = (g, g_1, g_2, g_3) = (g, g^{\alpha}, g^{\beta}, g^{\gamma}), sk = (\alpha, \beta, \gamma)$.

Sign: It chooses a random $r \in \mathbb{Z}_p$ and computes σ_m on $m \in \mathbb{Z}_p$

$$\sigma_m = \Big(r, \alpha + m\beta + r\gamma\Big).$$

Incorrect Proof. Given as input (g, g^a), $\mathcal{B}$ runs $\mathcal{A}$ and works as follows.

Setup. The simulator randomly chooses $x_1, y_1, x_2 \in \mathbb{Z}_p$ and sets

$$(\alpha, \beta, \gamma) = (a, x_1 a + y_1, a + x_2)$$

Query. The simulator uses $r = -a - x_1 m$ in computing the signature on m.

Partition: The signature on m^* with r^* is simulatable if $r^* = -a - x_1 m^*$. The adversary can compute (a, x_1) from the public key and queried signature.

Outline

1 Proof Structure

2 Simulatable and Reducible

3 Partition
- Partition:Standard
- Partition:Advanced

4 Summary

Partition: Advanced

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.

Good Partition: Example (1)

Suppose the adversary can forge a signature without signature query.

KeyGen: The key pair is $pk = (g, g^\alpha, g^\beta), \quad sk = (\alpha, \beta)$.
Sign: The signature on $m \in \mathbb{Z}_p$ is $\sigma_m = \left(r, g^{\frac{\beta - r}{\alpha - m}}\right)$. r is a random number.
Verify: The signature σ_m is valid if $e(\sigma_m, g^\alpha g^{-m}) = e(g^\beta g^{-r}, g)$.

Hard Problem: Given (g, g^a), it is hard to compute $(c, g^{\frac{1}{a+c}})$ for any $c \in \mathbb{Z}_p$.

Proof. Given $(g, g^a, g^{a^2}, \cdots, g^{a^q})$, the simulator chooses a random x and sets $pk = (g, g^\alpha, g^\beta) = (g, g^a, g^{a+w})$.

Question: Can this reduction work?

Good Partition: Example (1)

Answer is given in the next page.

Good Partition: Example (1)

Suppose the adversary can forge a signature without signature query.

KeyGen: The key pair is $pk = (g, g^\alpha, g^\beta), \quad sk = (\alpha, \beta)$.
Sign: The signature on $m \in \mathbb{Z}_p$ is $\sigma_m = \left(r, g^{\frac{\beta - r}{\alpha - m}}\right)$. r is a random number.
Verify: The signature σ_m is valid if $e(\sigma_m, g^\alpha g^{-m}) = e(g^\beta g^{-r}, g)$.

Hard Problem: Given (g, g^a), it is hard to compute $(c, g^{\frac{1}{a+c}})$ for any $c \in \mathbb{Z}_p$.

Proof. Given $(g, g^a, g^{a^2}, \cdots, g^{a^q})$, the simulator chooses a random x and sets $pk = (g, g^\alpha, g^\beta) = (g, g^a, g^{a+w})$.

Answer: No. The adversary can compute w and forge signature on m^* with $r^* = -(k-1)\cdot a + w + k \cdot m^*$ for any k.

Good Partition: Example (1)

Suppose the adversary can forge a signature without signature query.

KeyGen: The key pair is $pk = (g, g^\alpha, g^\beta), \quad sk = (\alpha, \beta)$.
Sign: The signature on $m \in \mathbb{Z}_p$ is $\sigma_m = \left(r, g^{\frac{\beta - r}{\alpha - m}}\right)$. r is a random number.
Verify: The signature σ_m is valid if $e(\sigma_m, g^\alpha g^{-m}) = e(g^\beta g^{-r}, g)$.

Hard Problem: Given (g, g^a), it is hard to compute $(c, g^{\frac{1}{a+c}})$ for any $c \in \mathbb{Z}_p$.

Proof. Given $(g, g^a, g^{a^2}, \cdots, g^{a^q})$, the simulator chooses a random x and sets $pk = (g, g^\alpha, g^\beta) = (g, g^a, g^{a \cdot w})$.

Question: Can this reduction work? Replacing $a + w$ with $a \cdot w$.

Good Partition: Example (1)

Answer is given in the next page.

Good Partition: Example (1)

Suppose the adversary can forge a signature without signature query.

KeyGen: The key pair is $pk = (g, g^{\alpha}, g^{\beta}), \quad sk = (\alpha, \beta)$.
Sign: The signature on $m \in \mathbb{Z}_p$ is $\sigma_m = \left(r, g^{\frac{\beta - r}{\alpha - m}}\right)$. r is a random number.
Verify: The signature σ_m is valid if $e(\sigma_m, g^{\alpha}g^{-m}) = e(g^{\beta}g^{-r}, g)$.

Hard Problem: Given (g, g^{a}), it is hard to compute $(c, g^{\frac{1}{a+c}})$ for any $c \in \mathbb{Z}_p$.

Proof. Given $(g, g^{a}, g^{a^2}, \cdots, g^{a^q})$, the simulator chooses a random x and sets $pk = (g, g^{\alpha}, g^{\beta}) = (g, g^{a}, g^{a \cdot w})$.

Answer: No. The adversary can compute w and forge signature on m^* with $r^* = -ka + w + (k + w)m^*$ for any k.

Good Partition: Example (1)

- Two simulation are introduced before, where the adversary knows the partition.
- The above two incorrect simulations can be combined together to obtain a correct simulation.
- The adversary will have no advantage in distinguishing which simulation is used.

Good Partition: Example (1)

Suppose the adversary can forge a signature without signature query.

KeyGen: The key pair is $pk = (g, g^\alpha, g^\beta), \quad sk = (\alpha, \beta)$.
Sign: The signature on $m \in \mathbb{Z}_p$ is $\sigma_m = \left(r, g^{\frac{\beta - r}{\alpha - m}}\right)$. r is a random number.
Verify: The signature σ_m is valid if $e(\sigma_m, g^\alpha g^{-m}) = e(g^\beta g^{-r}, g)$.

Hard Problem: Given (g, g^a), it is hard to compute $(c, g^{\frac{1}{a+c}})$ for any $c \in \mathbb{Z}_p$.

Proof. Given $(g, g^a, g^{a^2}, \cdots, g^{a^q})$, the simulator chooses a random bit $b \in \{0, 1\}$ and a random integer w, and sets

$$pk = (g, g^\alpha, g^\beta) = \begin{cases} (g, g^a, g^{a+w}) & b = 0 \\ (g, g^a, g^{a \cdot w}) & b = 1 \end{cases}$$

No answer is given. Try to analyze its correctness by yourself!

Outline

1 Proof Structure

2 Simulatable and Reducible

3 Partition
- Partition:Standard
- Partition:Advanced

4 **Summary**

Summary

A correct security reduction where the simulator doesn't know the secret key should satisfy the following conditions.

- The underlying hard problem is a computational hard problem.
- The simulator doesn't know the secret key.
- All queried signatures are simulatable without secret key.
- The simulation is indistinguishable from the real attack.
- The partition is intractable or indistinguishable.
- The forged signature is reducible.
- The advantage ϵ_R of solving hard problem is non-negligible.
- The time cost of the simulation is polynomial time.

Have a Try?

One-time signature where the adversary can query one signature only.

KeyGen: $pk = (g, g_1, g_2, g_3) = (g, g^{\alpha}, g^{\beta}, g^{\gamma}), sk = (\alpha, \beta, \gamma)$.

Sign: It chooses a random $r \in \mathbb{Z}_p$ and computes σ_m on $m \in \mathbb{Z}_p$

$$\sigma_m = \Big(r, \alpha + m\beta + r\gamma\Big).$$

Verify: The signature σ_m on m is valid if and only if

$$g^{\alpha + m\beta + r\gamma} = g_1 g_2^m g_3^r$$

Question: How to program a correct security reduction under the DL assumption? (The answer can be found in the book)


END


Introduction
to Security
Reduction