Security Proof

Outline

- Proof Components
- Random Oralce
- BLS Scheme
- ElGamal Scheme

Proof Components

A security proof by a security reduction should have the following components in order to prove that the proposed scheme is secure.

- **Simulation.** The reduction algorithm should show how the simulator generates a simulated scheme and interacts with the adversary.
- **Solution.** The reduction algorithm should show how the simulator solves the underlying hard problem by returning a solution to a problem instance with the help of the adversary's attack on the simulated scheme.
- **Analysis.** After the simulation and solution, there should be an analysis showing that the advantage of solving the underlying hard problem is non-negligible if the breaking assumption holds.

Proof Components

$$\varepsilon_R = P_S \cdot \varepsilon \cdot P_U$$

Digital Signatures

- **Simulation.** In this part, we show how the simulator uses the problem instance to generate a simulated scheme and interacts with the adversary following the unforgeability security model. If the simulator has to abort, the security reduction fails.
- **Solution.** In this part, we show how the simulator solves the underlying hard problem using the forged signature generated by the adversary. To be precise, the simulator should be able to extract a solution to the problem instance from the forged signature.
- **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.

Proof Components

此公式是正确的!

Encryption

$$\varepsilon_R = P_S(P_T - P_F)$$

- **Simulation.** In this part, we show how the simulator uses the problem instance (X, Z) to generate a simulated scheme and interacts with the adversary following the indistinguishability security model. Most importantly, the target Z in the problem instance must be embedded in the challenge ciphertext. If the simulator has to abort, it outputs a random guess of Z.
- **Solution.** In this part, we show how 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. The method of guessing Z is the same in all security reductions. To be precise, if $c' = c$, the simulator outputs that Z is true. Otherwise, $c' \neq c$, and it outputs that Z is false.
- **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.

Random Oracle

- A random oracle is typically used to represent an ideal hash function H whose outputs are random and uniformly distributed in its output space.

- Random oracles are very helpful for the simulator in programming security reductions. The reason is that the simulator can control and select any output that looks random and helps the simulator complete the simulation or force the adversary to launch a useful attack. Security proofs in the random oracle model are, therefore, believed to be much easier than those without random oracles.

Input	Hash Function	Output
x_1		y_1
x_2		y_2
x_3	$H(x_i) = y_i$	y_3
x_4		y_4
$\vdots$		$\vdots$

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

Random Oracle

The hash list created by the simulator is composed of input, output, and the corresponding state S. This hash list should satisfy the following conditions.

- The hash list is empty at the beginning before any hash queries are made.
- All tuples associated with queries will be added to this hash list.
- The secret state S must be unknown to the adversary.

For a security proof in the random oracle model, the simulator should add one more phase called H-Query (usually after the Setup phase) in the simulation to describe hash queries and responses. Note that this phase only appears in the security reduction, and it should not appear in the security model.

H-Query. The adversary makes hash queries in this phase. The simulator prepares a hash list $\mathscr{L}$ to record all queries and responses, where the hash list is empty at the beginning.

Digital Signature--Algorithm Definition

A digital signature scheme consists of the following four algorithms.

SysGen: The system parameter generation algorithm takes as input a security parameter λ. It returns the system parameters SP.

KeyGen: The key generation algorithm takes as input the system parameters SP. It returns a public/secret key pair (pk, sk).

Sign: The signing algorithm takes as input a message m from its message space, the secret key sk, and the system parameters SP. It returns a signature of m denoted by σ_m.

Verify: The verification algorithm takes as input a message-signature pair (m, σ_m), the public key pk, and the system parameters SP. It returns "accept" if σ_m is a valid signature of m signed with sk; otherwise, it returns "reject."

Correctness. Given any (pk, sk, m, σ_m), if σ_m is a valid signature of m signed with sk, the verification algorithm on (m, σ_m, pk) will return "accept."

Digital Signature--Security Model

The security model of existential unforgeability against chosen-message attacks (EU-CMA) can be described as follows.

Setup. Let SP be the system parameters. The challenger runs the key generation algorithm to generate a key pair (pk, sk) and sends pk to the adversary. The challenger keeps sk to respond to signature queries from the adversary.

Query. The adversary makes signature queries on messages that are adaptively chosen by the adversary itself. For a signature query on the message m_i, the challenger runs the signing algorithm to compute σ_{m_i} and then sends it to the adversary.

Forgery. The adversary returns a forged signature σ_{m^*} on some m^* and wins the game if

- σ_{m^*} is a valid signature of the message m^*.
- A signature of m^* has not been queried in the query phase.

The advantage ε of winning the game is the probability of returning a valid forged signature.

Definition **(EU-CMA)** *A signature scheme is* (t, q_s, ε)*-secure in the EU-CMA security model if there exists no adversary who can win the above game in time t with advantage* ε *after it has made* q_s *signature queries.*

BLS Scheme

SysGen: The system parameter generation algorithm takes as input a security parameter λ. It chooses a pairing group $\mathbb{PG} = (\mathbb{G}, \mathbb{G}_T, g, p, e)$, selects a cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$, and returns the system parameters $SP = (\mathbb{PG}, H)$.

KeyGen: The key generation algorithm takes as input the system parameters SP. It randomly chooses $\alpha \in \mathbb{Z}_p$, computes $h = g^{\alpha}$, and returns a public/secret key pair (pk, sk) as follows:

$$pk = h, \;\; sk = \alpha.$$

Sign: The signing algorithm takes as input a message $m \in \{0,1\}^*$, the secret key sk, and the system parameters SP. It returns the signature σ_m on m as

$$\sigma_m = H(m)^{\alpha}.$$

Verify: The verification algorithm takes as input a message-signature pair (m, σ_m), the public key pk, and the system parameters SP. It accepts the signature if

$$e(\sigma_m, \; g) = e\big(H(m), \; h\big).$$

Theorem 5.1.0.1 *Suppose the hash function H is a random oracle. If the CDH problem is hard, the BLS signature scheme is provably secure in the EU-CMA security model with reduction loss $L = q_H$, where q_H is the number of hash queries to the random oracle.*

Proof. Suppose there exists an adversary $\mathscr{A}$ who can (t, q_s, ε)-break the signature scheme in the EU-CMA security model. We construct a simulator $\mathscr{B}$ to solve the CDH problem. Given as input a problem instance (g, g^a, g^b) over the pairing group $\mathbb{PG}$, $\mathscr{B}$ controls the random oracle, runs $\mathscr{A}$, and works as follows.

Setup. Let $SP = \mathbb{PG}$ and H be the random oracle controlled by the simulator. $\mathscr{B}$ sets the public key as $h = g^a$, where the secret key α is equivalent to a. The public key is available from the problem instance.

H-Query. The adversary makes hash queries in this phase. Before receiving queries from the adversary, $\mathscr{B}$ randomly chooses an integer $i^* \in [1, q_H]$, where q_H denotes the number of hash queries to the random oracle. Then, $\mathscr{B}$ prepares a hash list to record all queries and responses as follows, where the hash list is empty at the beginning.

Let the i-th hash query be m_i. If m_i is already in the hash list, $\mathscr{B}$ responds to this query following the hash list. Otherwise, $\mathscr{B}$ randomly chooses w_i from $\mathbb{Z}_p$ and sets $H(m_i)$ as

$$H(m_i) = \begin{cases} g^{b+w_i} & \text{if } i = i^* \\ g^{w_i} & \text{otherwise} \end{cases}.$$

The simulator $\mathscr{B}$ responds to this query with $H(m_i)$ and adds $(i, m_i, w_i, H(m_i))$ to the hash list.

Query. The adversary makes signature queries in this phase. For a signature query on m_i, if m_i is the i^*-th queried message in the hash list, abort. Otherwise, we have $H(m_i) = g^{w_i}$.

$\mathscr{B}$ computes σ_{m_i} as

$$\sigma_{m_i} = (g^a)^{w_i}.$$

According to the signature definition and simulation, we have

$$\sigma_{m_i} = H(m_i)^{\alpha} = (g^{w_i})^a = (g^a)^{w_i}.$$

Therefore, σ_{m_i} is a valid signature of m_i.

Forgery. The adversary returns a forged signature σ_{m^*} on some m^* that has not been queried. If m^* is not the i^*-th queried message in the hash list, abort. Otherwise, we have $H(m^*) = g^{b+w_{i^*}}$.

According to the signature definition and simulation, we have

$$\sigma_{m^*} = H(m^*)^{\alpha} = (g^{b+w_{i^*}})^{a} = g^{ab+aw_{i^*}}.$$

The simulator $\mathscr{B}$ computes

$$\frac{\sigma_{m^*}}{(g^a)^{w_{i^*}}} = \frac{g^{ab+aw_{i^*}}}{(g^a)^{w_{i^*}}} = g^{ab}$$

as the solution to the CDH problem instance.

This completes the simulation and the solution. The correctness is analyzed as follows.

Indistinguishable simulation. The correctness of the simulation has been explained above. The randomness of the simulation includes all random numbers in the key generation and the responses to hash queries. They are

$$a, w_1, \cdots, w_{i^*-1}, b + w_{i^*}, w_{i^*+1}, \cdots, w_{q_H}.$$

According to the setting of the simulation, where a, b, w_i are randomly chosen, it is easy to see that they are random and independent from the point of view of the adversary. Therefore, the simulation is indistinguishable from the real attack.

Probability of successful simulation and useful attack. If the simulator successfully guesses i^*, all queried signatures are simulatable, and the forged signature is reducible because the message m_{i^*} cannot be chosen for a signature query, and it will be used for the signature forgery. Therefore, the probability of successful simulation and useful attack is $\frac{1}{q_H}$ for q_H queries.

Advantage and time cost. Suppose the adversary breaks the scheme with (t, q_s, ε) after making q_H queries to the random oracle. The advantage of solving the CDH problem is therefore $\frac{\varepsilon}{q_H}$. Let T_s denote the time cost of the simulation. We have $T_s = O(q_H + q_s)$, which is mainly dominated by the oracle response and the signature generation. Therefore, $\mathscr{B}$ will solve the CDH problem with $(t + T_s, \varepsilon/q_H)$.

This completes the proof of the theorem. □

PKE--Algorithm Definition

A public-key encryption scheme consists of the following four algorithms.

SysGen: The system parameter generation algorithm takes as input a security parameter λ. It returns the system parameters SP.

KeyGen: The key generation algorithm takes as input the system parameters SP. It returns a public/secret key pair (pk, sk).

Encrypt: The encryption algorithm takes as input a message m from its message space, the public key pk, and the system parameters SP. It returns a ciphertext $CT = E[SP, pk, m]$.

Decrypt: The decryption algorithm takes as input a ciphertext CT, the secret key sk, and the system parameters SP. It returns a message m or outputs $\perp$ to denote a failure.

Correctness. Given any (SP, pk, sk, m, CT), if $CT = E[SP, pk, m]$ is a ciphertext encrypted with pk on the message m, the decryption of CT with the secret key sk will return the message m.

PKE--Security Model

Formally, the security model of indistinguishability against chosen-ciphertext attacks (IND-CCA) can be described as follows.

Setup. Let SP be the system parameters. The challenger runs the key generation algorithm to generate a key pair (pk, sk) and sends pk to the adversary. The challenger keeps sk to respond to decryption queries from the adversary.

Phase 1. The adversary makes decryption queries on ciphertexts that are adaptively chosen by the adversary itself. For a decryption query on the ciphertext CT_i, the challenger runs the decryption algorithm and then sends the decryption result to the adversary.

Challenge. The adversary outputs two distinct messages m_0, m_1 from the same message space, which are adaptively chosen by the adversary itself. The challenger randomly chooses $c \in \{0,1\}$ and then computes a challenge ciphertext $CT^* = E[SP, pk, m_c]$, which is given to the adversary.

Phase 2. The challenger responds to decryption queries in the same way as in Phase 1 with the restriction that no decryption query is allowed on CT^*.

Guess. The adversary outputs a guess c' of c and wins the game if $c' = c$.

The advantage ε of the adversary in winning this game is defined as

$$\varepsilon = 2\left(\Pr[c' = c] - \frac{1}{2}\right).$$

Definition 2.2.0.1 (IND-CCA) *A public-key encryption scheme is (t, q_d, ε)-secure in the IND-CCA security model if there exists no adversary who can win the above game in time t with advantage ε after it has made q_d decryption queries.*

Definition 2.2.0.2 (IND-CPA) *A public-key encryption scheme is (t, ε)-secure in the security model of indistinguishability against chosen-plaintext attacks (IND-CPA) if the scheme is $(t, 0, \varepsilon)$-secure in the IND-CCA security model, where the adversary is not allowed to make any decryption query.*

ElGamal Scheme

SysGen: The system parameter generation algorithm takes as input a security parameter λ. It chooses a cyclic group $(\mathbb{G}, p, g)$ and returns the system parameters $SP = (\mathbb{G}, p, g)$.

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

$$pk = g_1, \quad sk = \alpha.$$

Encrypt: The encryption algorithm takes as input a message $m \in \mathbb{G}$, 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

$$CT = (C_1, C_2) = \left(g^r,\ g_1^r \cdot m\right).$$

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 the message by computing $C_2 \cdot C_1^{-\alpha} = g_1^r m \cdot (g^r)^{-\alpha} = m$.

ElGamal Scheme

SysGen: The system parameter generation algorithm takes as input a security parameter λ. It chooses a cyclic group $(\mathbb{G}, p, g)$ and returns the system parameters $SP = (\mathbb{G}, p, g)$.

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

$$pk = g_1, \quad sk = \alpha.$$

Encrypt: The encryption algorithm takes as input a message $m \in \mathbb{G}$, 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

$$CT = (C_1, C_2) = \left(g^r,\ g_1^r \cdot m\right).$$

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 the message by computing $C_2 \cdot C_1^{-\alpha} = g_1^r m \cdot (g^r)^{-\alpha} = m$.

Theorem 8.1.0.1 *If the DDH problem is hard, the ElGamal encryption scheme is provably secure in the IND-CPA security model with reduction loss $L = 2$.*

Proof. Suppose there exists an adversary $\mathscr{A}$ who can (t, ε)-break the encryption scheme in the IND-CPA security model. We construct a simulator $\mathscr{B}$ to solve the DDH problem. Given as input a problem instance (g, g^a, g^b, Z) over the cyclic group $(\mathbb{G}, g, p)$, $\mathscr{B}$ runs $\mathscr{A}$ and works as follows.

Setup. Let $SP = (\mathbb{G}, g, p)$. $\mathscr{B}$ sets the public key as $g_1 = g^a$ where $\alpha = a$. The public key is available from the problem instance.

Challenge. $\mathscr{A}$ outputs two distinct messages $m_0, m_1 \in \mathbb{G}$ to be challenged. The simulator randomly chooses $c \in \{0, 1\}$ and sets the challenge ciphertext CT^* as

$$CT^* = \left(g^b,\ Z \cdot m_c\right),$$

where g^b and Z are from the problem instance. Let $r = b$. If $Z = g^{ab}$, we have

$$CT^* = \left(g^b,\ Z \cdot m_c\right) = \left(g^r,\ g_1^r \cdot m_c\right).$$

Therefore, CT^* is a correct challenge ciphertext whose encrypted message is m_c.

Guess. $\mathscr{A}$ outputs a guess c' of c. The simulator outputs true if $c' = c$. Otherwise, false.

This completes the simulation and the solution. The correctness is analyzed as follows.

Indistinguishable simulation. The correctness of the simulation has been explained above. The randomness of the simulation includes all random numbers in the key generation and the challenge ciphertext generation. They are a in the secret key and b in the challenge ciphertext. According to the setting of the simulation, where a, b are randomly chosen, it is easy to see that the randomness property holds, and thus the simulation is indistinguishable from the real attack.

Probability of successful simulation. There is no abort in the simulation, and thus the probability of successful simulation is 1.

Probability of breaking the challenge ciphertext.

If Z is true, the simulation is indistinguishable from the real attack, and thus the adversary has probability $\frac{1}{2} + \frac{\varepsilon}{2}$ of guessing the encrypted message correctly.

If Z is false, it is easy to see that the challenge ciphertext is a one-time pad because the message is encrypted using Z, which is random and cannot be calculated from the other parameters given to the adversary. Therefore, the adversary only has probability $1/2$ of guessing the encrypted message correctly.

Advantage and time cost. The advantage of solving the DDH problem is

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

Let T_s denote the time cost of the simulation. We have $T_s = O(1)$. Therefore, the simulator $\mathscr{B}$ will solve the DDH problem with $(t + T_s, \varepsilon/2)$.

This completes the proof of the theorem. □