ELE2742 / ENG1467 - PUC-Rio, 2019.1

Lecture 6.5: Security Proof

DATE

Lecturer: Yi-Fan Tseng *Scribe: Yi-Fan Tseng*

1 Security Proof

A *security proof* is different from *security analysis*. A security analysis may only be specific to one or several attacks, or put limitations on the attacker's moves. However, there are always some attacks that have not been considered in the analysis. Some attacks may not exist now, but will be proposed in the future. In a security proof, we care only about

1. what goal the attacker wants to achieve;
2. what resources the attacker gets.

Note that we do not care how the attacker achieve its goal. What we need is to prove that the goal of the attacker can never be achieved. To prove it, we will adopt an powerful tool, called *reduction*. Reduction is an important concept in computability theory and complexity theory, which is an algorithm to transform a problem to another problem. Figure 1 shows an easy example. We first define two problems.

- $\mathcal{P}_1$: Given n numbers, find the maximum among the numbers.
- $\mathcal{P}_2$: Given n numbers, output n numbers with decreasing order.

Then there is a reduction from $\mathcal{P}_1$ to $\mathcal{P}_2$. Assume that we are able to make black-box access to an oracle $\mathcal{A}_2$, which is an oracle for "sorting algorithm". That is, we are allowed to input n numbers to the oracle $\mathcal{A}_2$, and receive n sorted numbers as the output form $\mathcal{A}_2$, while we do not know how $\mathcal{A}_2$ performs. To construct a reduction, we need to show that, given black-box access to $\mathcal{A}_2$, we can construct an algorithm $\mathcal{A}_1$ to solve problem $\mathcal{P}_1$. Figure 1 shows the reduction from $\mathcal{P}_1$ to $\mathcal{P}_2$, denoted by $\mathcal{P}_1 \propto \mathcal{P}_2$. Such reduction implies that "solving $\mathcal{P}_1$" cannot be harder than "solving $\mathcal{P}_2$".

When proving the security for a cryptographic scheme, we usually define a corresponding *security game* first. The security game should define

- the goal of the attacker;
- the resources accessed by the attacker;
- the operations the attacker is allowed to perform.

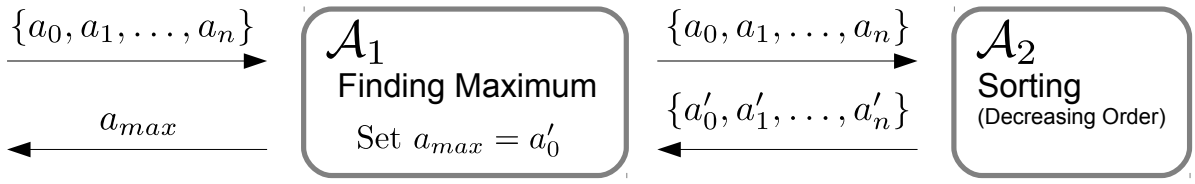

Figure 1: Reduction from "finding maximum" to "sorting"

The security game is designed to simulate what will happens in a real-world scheme, e.g., what is "breaking a scheme", "how an attacker interacts to other roles in the scheme", etc.

- Breaking the scheme $\Longleftrightarrow$ Winning the game
- The resources accessed by the attacker can be simulated by *oracles*.

To prove the security of our scheme, we need to construction a reduction from "solving a well-studied hard problem" to "winning the security game". That is, we need to prove that

if there is an attacker winning the security game, then we can construct an algorithm to solve a well-studied hard problem.

Since the hard problem is hard to be solved, and hence the scheme is hard to be broken. This type of security proof is known as *game-based* security proof.

In the security game (security model), the less restriction we put on the attacker or the more resource the attacker is allowed to obtain, then the attacker is of course more powerful. If we prove that for such type of attackers, the advantage of winning the security game is negligible, then the corresponding scheme is proven secure against such type of attacker. In a word, the stronger the attackers we assume in the game, the stronger security our scheme is able to achieve. In provable cryptography, we are used to calling the attacker as the "adversary".

2 Hard Problems, Complexity Assumptions

The hard problems or complexity assumptions used in security proofs are various. They could be hard problems in complexity theory, or the security of the underlying cryptographic primitives. We then introduce some well-studied hard problems. The hard problems can be roughly classified into two types, computational problems and decisional problems. The answer to a computational problem is usually a value (or a set of elements), and the answer to a decisional problem is a bit. A decisional problem is a yes-no problem.

Before introducing the hard problems, it is necessary to introduce the concept of "negligible".

Definition 2.1 (Negligible Function)**.** We say the a function $\mu : \mathbb{N} \to \mathbb{R}$ is a negligible function if, for any positive integer c, there exists an integer N_c, such that for all $x > N_c$,

$$|\mu(x)| < \frac{1}{x^c}.$$

We say that a probability (an advantage) is negligible if it is a negligible function in the security parameter λ. The concept of "non-negligible" is defined as a function which is *not negligible*.

2.1 Factorization-Based Problems

Definition 2.2 (Factorization Problem)**.** Given a composite N, output the factors of N.

For an algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathtt{Fac}}(\mathcal{A})$ for the factorization problem is defined as follows.

$$\mathsf{Adv}^{\mathtt{Fac}}(\mathcal{A}) = \mathbf{Pr}[\, p|N; p \leftarrow \mathcal{A}(N)\,].$$

Definition 2.3 (Factorization Assumption)**.** The *factorization assumption* holds if, for all probabilistic polynomial-time (PPT) algorithms $\mathcal{A}$, $\mathsf{Adv}^{\mathtt{Fac}}(\mathcal{A})$ is negligible.

Definition 2.4 (Quadratic Residuosity Problem)**.** Let $N = pq$ be the product of two large primes p, q. Let x be an integer such that the Jacobi symbol $J\left(\frac{x}{N}\right) = 1$. The *quadratic residuosity problem* is, given (x, N), to decide whether $x \in QR_N$. Without loss of generality, we may assume that for an algorithm $\mathcal{A}$ to solve the quadratic residuosity problem, $\mathcal{A}(x, N) = 1$ if the algorithm $\mathcal{A}$ "think" that $x \in QR_N$, and $\mathcal{A}(x, N) = 0$ if the algorithm $\mathcal{A}$ "think" that $x \in QNR_N$.

We next define the advantage for an algorithm $\mathcal{A}$ in solving the quadratic residuosity problem.

$$\mathsf{Adv}^{\mathtt{QR}}(\mathcal{A}) = \left|\mathbf{Pr}[\mathcal{A}(x, N) = 1; x \in QR_N] - \mathbf{Pr}[\mathcal{A}(x, N) = 1; x \in QNR_N]\right|,$$

where $J\left(\frac{x}{N}\right) = 1$.

Definition 2.5 (Quadratic Residuosity Assumption)**.** The *quadratic residuosity assumption* holds if, for all PPT algorithms $\mathcal{A}$, $\mathsf{Adv}^{\mathtt{QR}}(\mathcal{A})$ is negligible.

2.2 Discrete-Logarithm-Based Problems

Let $\mathbb{G}$ be a cyclic group with prime order p and a generator g.

Definition 2.6 (Discrete-Log Problem)**.** Given $(p, g, y \in \mathbb{G})$, find $x = \log_g(y)$.

For an algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathtt{DL}}(\mathcal{A})$ for the DL problem is defined as follows.

$$\mathsf{Adv}^{\mathtt{DL}}(\mathcal{A}) = \mathbf{Pr}[\, y = g^x; x \leftarrow \mathcal{A}(p, g, y)\,].$$

Definition 2.7 (Discrete-Log Assumption)**.** The *discrete-log assumption* holds if, for all PPT algorithms $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{DL}}(\mathcal{A})$ is negligible.

Definition 2.8 (Computational Diffie-Hellman Problem)**.** Let $X = g^x, Y = g^y$ for some $x, y \in \mathbb{Z}_p$. The computational Diffie-Hellman problem is, given (p, g, X, Y), to compute $Z = g^{xy}$.

For an algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathrm{CDH}}(\mathcal{A})$ for the CDH problem is defined as follows.

$$\mathsf{Adv}^{\mathrm{CDH}}(\mathcal{A}) = \mathbf{Pr}[\, Z = g^{xy}; Z \leftarrow \mathcal{A}(p, g, X, Y)\,].$$

Definition 2.9 (Computational Diffie-Hellman Assumption)**.** The *CDH assumption* holds if, for all PPT algorithms $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{CDH}}(\mathcal{A})$ is negligible.

Definition 2.10 (Desisional Diffie-Hellman Problem)**.** Let $X = g^x, Y = g^y$ for some $x, y \in \mathbb{Z}_p$. The desicional Diffie-Hellman problem is, given (p, g, X, Y, Z), to decide whether $Z \stackrel{?}{=} g^{xy}$.

For an algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathrm{DDH}}(\mathcal{A})$ for the DDH problem is defined as follows.

$$\mathsf{Adv}^{\mathrm{DDH}}(\mathcal{A}) = \left|\mathbf{Pr}[\mathcal{A}(p, g, X, Y, Z) = 1; Z = g^{xy}] - \mathbf{Pr}[\mathcal{A}(p, g, X, Y, Z) = 1; Z = g^z, z \stackrel{\$}{\leftarrow} \mathbb{Z}_p]\right|.$$

Definition 2.11 (Desicional Diffie-Hellman Assumption)**.** The *DDH assumption* holds if, for all PPT algorithms $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{DDH}}(\mathcal{A})$ is negligible.

2.3 Pairing-Based Problems

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map, where $\mathbb{G}, \mathbb{G}_T$ are two cyclic groups with prime order p. Let g be a generator of $\mathbb{G}$. Note that the DDH problem is easy in paring groups. For a DDH instance (p, g, X, Y, Z), it is easy to verify that

$$Z = g^{xy} \iff e(X, Y) = e(g, Z).$$

Definition 2.12 (Computational Bilinear Diffie-Hellman Problem)**.** Let $X = g^x, Y = g^y, Z = g^z$ for some $x, y, z \in \mathbb{Z}_p$. The CBDH problem is, given (p, g, X, Y, Z), to compute $T = e(g, g)^{xyz}$.

For an algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathrm{CBDH}}(\mathcal{A})$ for the CBDH problem is defined as follows.

$$\mathsf{Adv}^{\mathrm{CBDH}}(\mathcal{A}) = \mathbf{Pr}[\, T = e(g, g)^{xyz}; T \leftarrow \mathcal{A}(p, g, X, Y, Z)\,].$$

Definition 2.13 (Computational Bilinear Diffie-Hellman Assumption)**.** The *CBDH assumption* holds if, for all PPT algorithms $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{CBDH}}(\mathcal{A})$ is negligible.

Definition 2.14 (Decisional Bilinear Diffie-Hellman Problem)**.** Let $X = g^x, Y = g^y, Z = g^z$ for some $x, y, z \in \mathbb{Z}_p$. The DBDH problem is, given (p, g, X, Y, Z, T), to decide whether $T \stackrel{?}{=} e(g, g)^{xyz}$.

For an algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathrm{DBDH}}(\mathcal{A})$ for the DBDH problem is defined as follows.

$$\mathsf{Adv}^{\mathrm{DBDH}}(\mathcal{A}) = \left| \begin{array}{ll} & \mathbf{Pr}[\mathcal{A}(p, g, X, Y, Z, T) = 1; T = e(g, g)^{xyz}] \\ - & \mathbf{Pr}[\mathcal{A}(p, g, X, Y, Z, T) = 1; T = e(g, g)^{t}, t \xleftarrow{\$} \mathbb{Z}_p] \end{array} \right|.$$

Definition 2.15 (Decisional Bilinear Diffie-Hellman Assumption)**.** The *DBDH assumption* holds if, for all PPT algorithms $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{DBDH}}(\mathcal{A})$ is negligible.

3 Security Model and Proof for Key Exchange Protocol

In this section, we briefly introduce the security model for key exchange protocol, and demonstrate the security proofs for Deffie-Hellmand key exchange protocol and Joux's one-round 3-party protocol. We refer the reader to the paper of Canetti and Krawczyk [4] for more details.

For a secure key exchange protocol, it is desirable that the session key should be only computed by the two parties in the communication, and should not be computed by the adversary from what it can eavesdrop. In other word, the *goal* of the adversary is to recover the session key. However, under some circumstance, the adversary need not to recover the entire session key, but only need to learn the information of the session key. Therefore, we can slightly modify the *goal* of the adversary from "recover the entire session key" to "distinguish a true session key from a randomly chosen key". It is inspired from that, if the adversary is able to obtain the information of the session key from what it eavesdrop, then it should be able to tell a true key from a false one. Note that this is an easier goal for the adversary, since if the adversary is able to recover the entire session key, then it must be able to tell a true on from a false one.

We next discuss the *resources* the adversary is allowed to obtain. In a key exchange protocol, the adversary is able to obtain

1. the public parameter;
2. the communications it eavesdrops.

Now, after analyzing the *goal* and the *resources* of the adversary, we can design the security game for a key exchange protocol. The game contains three phases, **Setup** phase, **Test** phase, and **Guess** phase. There are two roles involved in the security game, the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$. The challenger is responsible for handling the game to play with the adversary.

Setup. The challenger $\mathcal{C}$ computes and sends the public parameter to the adversary $\mathcal{A}$.

Test. The challenger $\mathcal{C}$ simulates the communication of the key exchange procedure, and sends to the adversary $\mathcal{A}$ what it should be able to eavesdrop. Let K_1 be the valid key generated from the key exchange procedure. The challenger next chooses a random key K_0, and chooses a random bit

$b \xleftarrow{\$} \{0,1\}$. Finally, K_b is sent to the adversary as well.

Guess. The adversary outputs a bit b'.

This game is sometimes called the *session key* security game [4]. The adversary $\mathcal{A}$ is said to win the game if $b' = b$. We next define the advantage $\mathsf{Adv}^{\mathtt{sk}}(\mathcal{A})$ of an adversary winning the above game.

$$\mathsf{Adv}^{\mathtt{sk}}(\mathcal{A}) = \left| \mathbf{Pr}[b' = b] - \frac{1}{2} \right|$$

A key exchange protocol achieves session key security if, for all PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathtt{sk}}(\mathcal{A})$ is negligible.

3.1 The Security Proof for Deffie-Hellamn Key Exchange Protocol

In this section, we will prove that Diffie-Hellman key exchange protocol achieves session key security if the decisional Diffie-Helmman assumption holds.

We first recall the protocol (Figure 2) and the assumption. Let p, q be two large primes such that $q|p-1$. Let $\mathbb{G}$ be a subgroup of $\mathbb{Z}_p^*$ with order q, and g generate $\mathbb{G}$.

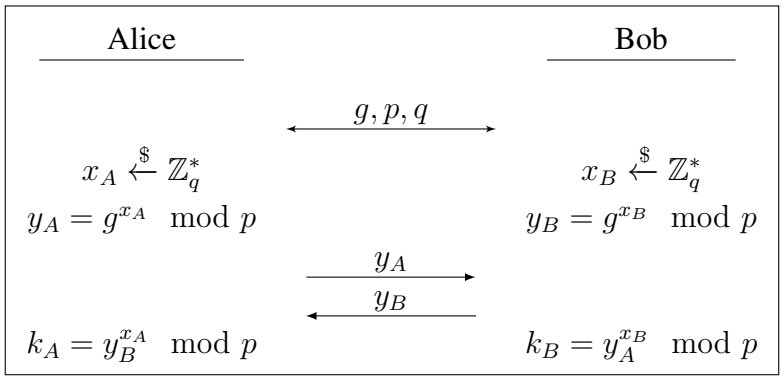

Figure 2: Deffie-Hellman Key Exchange Protocol

Definition 3.1 (The DDH problem over group $\mathbb{G}$)**.** Given $(g, p, q, X = g^x \mod p, Y = g^y \mod p, Z)$, decide whether $Z \stackrel{?}{=} g^{xy} \mod p$.

Theorem 3.2. *Diffie-Hellman key exchange protocol satisfies the session key security if the DDH problem over group $\mathbb{G}$ is hard.*

Proof. Assume that there is an adversary $\mathcal{A}$ wins the session key security game, then we can construct an algorithm $\mathcal{C}$ to solve the DDH problem over $\mathbb{G}$. Given the instance (g, p, q, X, Y) of the DDH problem, the challenger $\mathcal{C}$ construct the session key security game played with $\mathcal{A}$ as follows.

Setup. $\mathcal{C}$ sends the public parameter (g, p, q) to $\mathcal{A}$.

Test. $\mathcal{C}$ sets $y_A \leftarrow X, y_B \leftarrow Y$, and sets the session key $k \leftarrow Z$.

Guess. $\mathcal{A}$ outputs a bit b'.

After receiving the adversary's output, the challenger $\mathcal{C}$ outputs b'.

Correctness of the Game. The above game is indistinguishable from the game described in the previous section. From $\mathcal{A}$'s view, y_A, y_B are random element in $\mathbb{G}$. When $Z = g^{xy} \mod p$, it is the case that K_1 is sent to $\mathcal{A}$. When Z is a random element in $\mathbb{G}$, it is the case that K_0 is sent to $\mathcal{A}$.

Advantage of breaking the DDH Assumption. Assume that the adversary wins the session key security game with non-negligible advantage

$$\mathsf{Adv}^{\mathrm{sk}}(\mathcal{A}) = \left| \mathbf{Pr}[b' = b] - \frac{1}{2} \right|.$$

$$\implies Pr[b' = b] = \mathsf{Adv}^{\mathrm{sk}}(\mathcal{A}) + \frac{1}{2}$$

We then use $\mathcal{A}$ as a subroutine to solve the DDH problem. When $Z = g^{xy} \mod p$, it is equivalent to that $\mathcal{C}$ give K_1 to $\mathcal{A}$, and thus $\mathcal{A}$ outputs 1 with probability $\mathsf{Adv}^{\mathrm{sk}}(\mathcal{A}) + \frac{1}{2}$, so does $\mathcal{C}$. I.e.,

$$\mathbf{Pr}[\mathcal{C}(g, p, q, X, Y, Z = g^{xy}) = 1] = \mathsf{Adv}^{\mathrm{sk}}(\mathcal{A}) + \frac{1}{2}.$$

When $Z \xleftarrow{\$} \mathbb{G}$, it is equivalent to that $\mathcal{C}$ give K_0 to $\mathcal{A}$, and thus $\mathcal{A}$ outputs 1 with probability $1 - (\mathsf{Adv}^{\mathrm{sk}}(\mathcal{A}) + \frac{1}{2}) = \frac{1}{2} - \mathsf{Adv}^{\mathrm{sk}}(\mathcal{A})$, so does $\mathcal{C}$. I.e.,

$$\mathbf{Pr}[\mathcal{C}(g, p, q, X, Y, Z \xleftarrow{\$} \mathbb{G}) = 1] = \frac{1}{2} - \mathsf{Adv}^{\mathrm{sk}}(\mathcal{A}).$$

Therefore, the advantage of the challenger $\mathcal{C}$ in breaking the DDH assumption is

$$\begin{aligned} & \mathsf{Adv}^{\mathrm{DDH}}(\mathcal{C}) \\ = & \left| \mathbf{Pr}[\mathcal{C}(g, p, q, X, Y, Z = g^{xy}) = 1] - \mathbf{Pr}[\mathcal{C}(g, p, q, X, Y, Z \xleftarrow{\$} \mathbb{G}) = 1] \right| \\ = & \left| (\mathsf{Adv}^{\mathrm{sk}}(\mathcal{A}) + \tfrac{1}{2}) - (\tfrac{1}{2} - \mathsf{Adv}^{\mathrm{sk}}(\mathcal{A})) \right| \\ = & \ 2 \cdot \mathsf{Adv}^{\mathrm{sk}}(\mathcal{A}). \end{aligned}$$

That means the challenger $\mathcal{C}$ also breaking the DDH assumption with non-negligible advantage. □

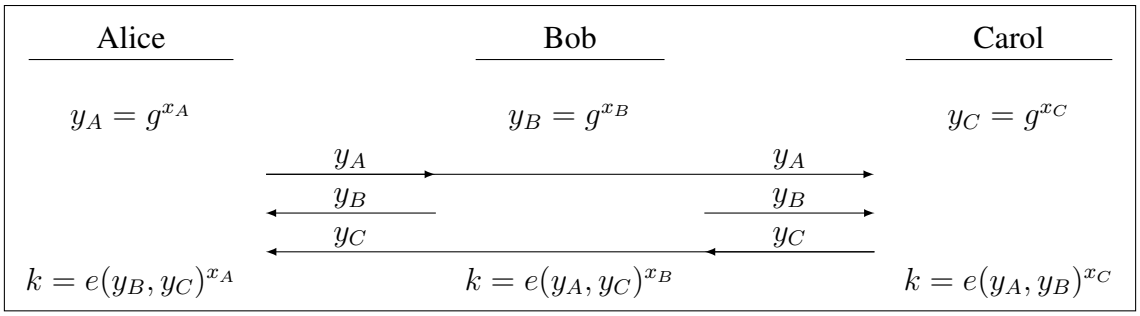

Figure 3: Joux's Key Exchange Protocol

3.2 The Security Proof of Joux's Key Exchange Protocol

In this section, we will prove that Joux's protocol is secure based on the DBDH assumption. We first recall Joux's Key Exchange Protocol (Figure 3) and the DHDH assumption.

Definition 3.3 (The DBDH Assumption)**.** Given $(g, X = g^x, Y = g^y, Z = g^z, T)$, decide whether $T \stackrel{?}{=} e(g, g)^{xyz}$.

Theorem 3.4. *Joux's key exchange protocol achieves session key security if the DBDH assumption holds.*

Proof. We only give the sketch of the proof due to its similarity to the proof of Theorem 3.2. The challenger sets $y_A \leftarrow X, y_B \leftarrow Y, y_C \leftarrow Z$, and sets the session key $k \leftarrow T$. Then the challenger sends (y_A, y_B, y_C, k) to the adversary. Finally, the challenger outputs what the adversary outputs. The challenger breaks the DBDH assumption with the advantage within a constant factor of that of the adversary winning the game. □

4 Security Model and Proof for Public Encryption

As we stated in Section 3, the security model of a public key encryption can be defined by the *goal* and the *resources* of the adversary. In a secure encryption scheme, a ciphertext should reveal no information about its message, unless the corresponding private key is given. Note that the case is very similar to the session key security. We can classify the adversary's goal into two types.

- One-Wayness: The adversary tries to recover the message of a given ciphertext without the knowledge of the corresponding private key.
- Indistinguishability: Given a ciphertext, which is the encryption of one of the two massage chosen by the adversary, the adversary tries to tell which of the two messages is encrypted.

It is easy to see that indistinguishability is an easier goal to the adversary. Actually, one-wayness is not a standard security notion for encryption in modern cryptography. Thus, in this lecture we

only discuss indistinguishability.

We next discuss the resources the adversary is allowed to access. The public parameter, including the public key, is of course accessible to the adversary. Then we discuss two attack models for encryption.

- Chosen-Plaintext Attacks (CPA): The adversary is allowed to obtain the ciphertext for the plaintext of its choosing.
- Chosen-Ciphertext Attacks (CCA): The adversary is allowed to obtain the plaintext for the ciphertext of its choosing.

For a public key encryption scheme, since the adversary is able to access the public key, the chosen-plaintext attacks can be easily performed by the adversary. However, to decrypt a ciphertext needs the knowledge of the private key. Therefore, to model the CCA, it is suffice to provide a *decryption oracle* in the security game.

We then introduce the general security game for public key encryption below, called *IND-ATK* game, where the term *ATK* depends on the attack model.

Setup. The challenger sends the public parameter, including the public key PK to the adversary.

Phase 1. The adversary is allowed to make polynomially many queries to the oracles provided by the challenger.

Challenge. The adversary submits two distinct messages $(\mathsf{M}_0, \mathsf{M}_1)$ of the same length to the challenger. Then the challenger chooses a bit $b \xleftarrow{\$} \{0,1\}$, and computes $\mathsf{CT}^* \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \mathsf{M}_b)$. Finally, the challenge ciphertext CT^* is sent to the adversary.

Phase 2. The adversary is allowed to make polynomially many queries to the oracles provided by the challenger, as the same as **Phase 1**, except for some natural (minimum) restrictions.

Guess. The adversary outputs a bit b'.

The adversary wins the game if $b' = b$.

- If ATK = CPA: In **Phase 1** and **Phase 2**, the adversary access no oracles, since it can encrypt any plaintext itself.
- If ATK = CCA: In both **Phase 1** and **Phase 2**, the adversary is able to make queries to a *decryption oracle*. The restriction is that the adversary is not allowed to make a decryption query with CT^* in **Phase 2**, in order to prevent the trivial way to win the IND-CCA game.

The advantage of the adversary $\mathcal{A}$ of winning the IND-ATK game is defined as

$$\mathsf{Adv}^{\text{IND-ATK}}(\mathcal{A}) = \left| \mathbf{Pr}[b' = b] - \frac{1}{2} \right|.$$

There are various security notions for public key encryption. We refer the readers to [1] for more details.

4.1 The Insecurity of RSA Encryption under Chosen-Plaintext Attacks

Though RSA encryption is the most-widely encrypting technology nowadays, the original version of RSA encryption is not IND-CPA secure.

Theorem 4.1. *RSA encryption is not IND-CPA secure.*

Proof. To prove the theorem, it is suffice to construct an adversary $\mathcal{A}$ to win the IND-CPA game with non-negligible advantage. After receiving the public parameter (e, N), the adversary $\mathcal{A}$ chooses two distinct messages $(\mathsf{M}_0, \mathsf{M}_1)$ and sends them to the challenger. Upon receiving the challenge ciphertext CT^*, the adversary $\mathcal{A}$ performs as follows.

1. Compute $\mathsf{CT}_0 = \mathsf{M}_0^e \pmod N$ and $\mathsf{CT}_1 = \mathsf{M}_1^e \pmod N$.
2. Output 0 if $\mathsf{CT}^* = \mathsf{CT}_0$; output 1 if $\mathsf{CT}^* = \mathsf{CT}_1$; otherwise, output a random bit $b' \xleftarrow{\$} \{0, 1\}$.

It is easy to verify that $\mathcal{A}$ wins the IND-CPA game with non-negligible advantage. □

Actually, the deterministic encryption schemes, such as RSA encryption and Rabin encryption, are not able to achieve IND-CPA security. In practice, we apply random padding, such as OAEP [3], to make encryption probabilistic.

4.2 Security Proofs for ElGamal Encryption

We first recall the details of ElGamal encryption scheme.

$\mathsf{KeyGen}(1^\lambda)$: Taking as input the security parameter 1^λ, the algorithm performs as follows.

1. Choose two large primes p, q, such that $q|p-1$. Let $\mathbb{G}$ be a cyclic group with order q and a generator g.
2. Choose $x \xleftarrow{\$} \mathbb{Z}_q^*$.
3. Compute $y = g^x \mod p$.
4. Output $\mathsf{PK} = (g, y, p, q)$ as the public key, $\mathsf{SK} = x$ as the private key.

$\mathsf{Encrypt}(\mathsf{PK}, \mathsf{M})$: Taking as inputs the public key $\mathsf{PK} = (g, y, p, q)$ and a message $\mathsf{M} \in \mathbb{Z}_N^*$, the algorithm performs as follows

1. Choose $r \xleftarrow{\$} \mathbb{Z}_q^*$.
2. Compute $C_1 = g^r \mod p$.
3. Compute $C_2 = \mathsf{M} \cdot y^r \mod p$.
4. Output $\mathsf{CT} = (C_1, C_2)$.

$\mathsf{Decyrpt}(\mathsf{PK}, \mathsf{SK}, \mathsf{CT})$: Taking as inputs the public key $\mathsf{PK} = (g, y, p, q)$, the private key $\mathsf{SK} = x$, and the ciphertext $\mathsf{CT} = (C_1, C_2)$, the algorithm outputs the message

$$\mathsf{M} = \frac{C_2}{C_1^x} \mod p.$$

Theorem 4.2. *ElGamal encryption is IND-CPA secure if the DDH assumption over $\mathbb{G}$ (Definition 3.1) holds.*

Proof. Given the instance of DDH problem (g, p, q, X, Y, Z), we construct a challenger to play an IND-CPA game with the adversary $\mathcal{A}$ as follows.

Setup. $\mathcal{C}$ sets the public key $\mathsf{PK} \leftarrow (g, X, p, q)$, which means we implicitly set the private key to x. The public key PK is sent to $\mathcal{A}$.

Challenge. After receiving two distinct messages $(\mathsf{M}_0, \mathsf{M}_1)$ from $\mathcal{A}$, $\mathcal{C}$ performs as follows.

1. Select $b \xleftarrow{\$} \{0, 1\}$.
2. Set $C_1 \leftarrow Y$, which means we implicitly set the randomness used in encryption to y.
3. Compute $C_2 = \mathsf{M}_b \cdot Z \mod p$.
4. Return the challenge ciphertext $\mathsf{CT}^* = (C_1, C_2)$.

Guess. The adversary outputs a bit b'.

The challenger then outputs 1 if $b' = b$; otherwise outputs 0.

If $Z = g^{xy} \mod p$, then we have that $C_1 = g^y, C_2 = \mathsf{M}_b \cdot X^y$, and thus CT^* is a valid ciphertext. Let the advantage of $\mathcal{A}$ be

$$\mathsf{Adv}^{\text{IND-CPA}}(\mathcal{A}) = \left| \mathbf{Pr}[b' = b] - \frac{1}{2} \right|.$$

Therefore, we have that

$$\mathbf{Pr}[\mathcal{C}(g, p, q, X, Y, Z = g^{xy}) = 1] = \mathsf{Adv}^{\text{IND-CPA}}(\mathcal{A}) + \frac{1}{2}.$$

If Z is a random element in $\mathbb{G}$, then the message M_b is completely hidden from the adversary (since it is a one-time pad), and thus the advantage of $\mathcal{A}$ must be 0, i.e.

$$\mathbf{Pr}[b' = b] = \frac{1}{2}.$$

Therefore, we have that

$$\mathbf{Pr}[\mathcal{C}(g, p, q, X, Y, Z \xleftarrow{\$} \mathbb{G}) = 1] = \frac{1}{2}.$$

Finally, we have that

$$\begin{aligned} & \mathsf{Adv}^{\mathtt{DDH}}(\mathcal{C}) \\ = & \left|\mathbf{Pr}[\mathcal{C}(g, p, q, X, Y, Z = g^{xy}) = 1] - \mathbf{Pr}[\mathcal{C}(g, p, q, X, Y, Z \xleftarrow{\$} \mathbb{G}) = 1]\right| \\ = & \left|(\mathsf{Adv}^{\mathtt{IND\text{-}CPA}}(\mathcal{A}) + \tfrac{1}{2}) - (\tfrac{1}{2})\right| \\ = & \mathsf{Adv}^{\mathtt{IND\text{-}CPA}}(\mathcal{A}). \end{aligned}$$

□

Theorem 4.3. *ElGamal encryption is not IND-CCA secure.*

Proof. We construct an adversary $\mathcal{A}$ to win the IND-CCA game below. After receiving the challenge ciphertext $\mathsf{CT}^* = (C_1, C_2)$, $\mathcal{A}$ performs as follows.

1. Randomly choose a message M'.
2. Compute the ciphertext $(C_1', C_2') \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \mathsf{M}')$.
3. Compute $\mathsf{CT}'' = (C_1 \cdot C_1', C_2 \cdot C_2')$.
4. Query the decryption oracle with CT'' and obtain the plaintext M''.
5. If $\mathsf{M}'' = \mathsf{M}_0 \cdot \mathsf{M}'$, output 0; otherwise output 1.

We then explain why the attack works. Let

$$\begin{array}{ll} C_1 = g^r & C_2 = \mathsf{M}_b \cdot y^r \\ C_1' = g^{r'} & C_2' = \mathsf{M}' \cdot y^{r'} \end{array}$$

We have

$$\begin{aligned} C_1'' &= C_1 \cdot C_1' = g^r \cdot g^{r'} = g^{r+r'} \\ C_2'' &= C_2 \cdot C_2' = (\mathsf{M}_b \cdot g^r) \cdot (\mathsf{M}' \cdot g^{r'}) = (\mathsf{M}_b\mathsf{M}') \cdot g^{r+r'}. \end{aligned}$$

Therefore, CT'' is a valid encryption of $(\mathsf{M}_b\mathsf{M}')$ under randomness $(r + r')$. By querying the decryption oracle to obtain $(\mathsf{M}_b\mathsf{M}')$, the adversary is able to find out whether $b = 0$ or $b = 1$. □

Observe that in ElGamal encryption scheme, we can somehow manipulate the the encrypted message to generate another ciphertext. The encryption scheme with this property is called *homomorphic encryption*. It is easy to see that a homomorphic encryption cannot be proven IND-CCA secure.

5 Security Model and Proof for Authentication Protocol

In an authentication protocol, the two parties, say the server and the client, will first share some secrets. Now, assume that the server and the client now are performing an authentication protocol. From the server's view, after authenticating, it will be able to ensure the other party is the client. It is the similar case from the client's view.

The security model for authentication is first formalized by Bellare and Rogaway [2], and thus called the BR model. We begin by introducing two concepts.

- Matching: For a party, the other party whom it communicates with is the specific partner who has shared a secret with the party.
- Acceptance: From a party's view, the other party pass the verification.

Then for a secure authentication protocol, the correctness can be defined by

$$\text{Matching} \implies \text{Acceptance},$$

and the security can be defined by

$$\text{Acceptance} \implies \text{Matching}.$$

5.1 Security Proof for the Nonce-Based Authentication

We first recall that the nonce-based authentication (Figure 6) shown in Chapter 4. In Chapter 4, we have already analyzed the security of the protocol based on the security of the underlying symmetric encryption. Therefore, we needs to prove that, if there is an adversary violating the security of the authentication protocol, then we can construct an algorithm to break the underlying symmetric encryption. That is to say, the hard problem we used in the proof is "the hardness of breaking the underlying symmetric encryption". We then introduce the security game for symmetric encryption.

IND-CCA Game for Symmetric Encryption

Setup. The challenger sends the public parameter to the adversary. Note that the encryption key k is not sent to the adversary.

Phase 1. The adversary is allowed to make polynomially many queries to an *encryption oracle* and a *decryption oracle*.

Challenge. The adversary submits two messages $(\mathsf{M}_0, \mathsf{M}_1)$ with the same length to the challenger. Then the challenger chooses $b \xleftarrow{\$} \{0, 1\}$, and computes $\mathsf{CT}^* \leftarrow E_k(\mathsf{M}_b)$.

Phase 2. The adversary is allowed to make polynomially many queries to an *encryption oracle* and a *decryption oracle*, with the restriction that the adversary is not allowed to query the decryption

oracle with CT^*.

Guess. The adversary outputs a bit b'.

The advantage of the adversary of winning the IND-CCA game is defined as

$$\mathsf{Adv}^{\text{IND-CCA}}(\mathcal{A}) = \left| \mathbf{Pr}[b' = b] - \frac{1}{2} \right|.$$

Since in a symmetric encryption scheme, encrypting a message needs the encryption key. Hence the challenger needs to provide an *encryption oracle* to the adversary.

Theorem 5.1. *The protocol shown in Figure 6 is a secure mutual authentication protocol if the underlying symmetric encryption scheme is IND-CCA secure.*

Proof. To prove the theorem, we need to prove that if there is an adversary breaking the authentication protocol, then we can construct an algorithm to break the underlying symmetric encryption scheme. Note that, the challenger of the authentication game is as well the adversary of the IND-CCA game. In order to prevent the reader from confusing, we denote the adversary/challenger of the authentication game as $\mathcal{A}_{auth}/\mathcal{C}_{auth}$, and the adversary/challenger of the IND-CCA game for symmetric encryption as $\mathcal{A}_{se}/\mathcal{C}_{se}$.

The proof is divided into two parts. We need to prove that

- if a party passes the verification of the client's side, then the party must be the server,
- if a party passes the verification of the server's side, then the party must be the client.

Lemma 5.2. *There is no adversary being able to impersonate the server if the underlying symmetric encryption is IND-CCA secure.*

Proof. If there is an adversary $\mathcal{A}_{auth}$ being able to impersonate the server to pass the verification, then we can construct an algorithm $\mathcal{C}_{auth}$ ($\mathcal{A}_{se}$) to break the underlying symmetric encryption scheme. The detail of the proof is demonstrated in Figure 4. Since $t_1 = \mathsf{CT}^*$ must be different from t_2, $\mathcal{C}_{auth}$ is able to make a decryption query to $\mathcal{C}_{se}$ with t_2. Besides, we have assume that the adversary $\mathcal{A}_{auth}$ is able to pass the verification with non-negligible advantage, thus the term r''_C in the return of the decryption query must be $r_C^{(0)} + 1$ or $r_C^{(1)} + 1$ with the same advantage. □

Lemma 5.3. *There is no adversary being able to impersonate the client if the underlying symmetric encryption is IND-CCA secure.*

Proof. If there is an adversary $\mathcal{A}_{auth}$ being able to impersonate the client to pass the verification, then we can construct an algorithm $\mathcal{C}_{auth}$ ($\mathcal{A}_{se}$) to break the underlying symmetric encryption scheme. The detail of the proof is demonstrated in Figure 5. Since $t_3 = \mathsf{CT}^*$ must be different from t_2, $\mathcal{C}_{auth}$ is able to make a decryption query to $\mathcal{C}_{se}$ with t_3. Besides, we have assume that the adversary $\mathcal{A}_{auth}$ is able to pass the verification with non-negligible advantage, thus the term r''_S in the return of the decryption query must be $r_S^{(0)} + 1$ or $r_S^{(1)} + 1$ with the same advantage. □

Lemma 5.2 and Lemma 5.3 complete the proof of Theorem 5.1. □

The authentication protocol also allows two parties to establish a session key, and hence it is also a secure key exchange protocol. We then prove the session key security for the authentication scheme.

Theorem 5.4. *The authentication protocol shown in Figure 6 satisfies the session key security if the underlying symmetric encryption is IND-CCA secure.*

Proof. We give the sketch of the proof. In the session key security game, the challenger $\mathcal{C}$ needs to simulate the view of the adversary $\mathcal{A}$, i.e., (t_1, t_2, t_3). The challenger $\mathcal{C}$ performs as follows.

1. Choose three random numbers r, r_C, r_S.
2. Invoke the **Challenge** phase of the IND-CCA game with (r, r_C) to obtain the challenge ciphertext CT^*.
3. Query the encryption oracle with $(r_S \| r_C + 1)$ to obtain the ciphertext CT_2.
4. Query the encryption oracle with $r_S + 1$ to obtain the ciphertext CT_3.
5. Set $t_1 \leftarrow \mathsf{CT}^*, t_2 \leftarrow \mathsf{CT}_2, t_3 \leftarrow \mathsf{CT}_3$.
6. Compute the session key $k = H(r_C, r_S)$.
7. Finally, (t_1, t_2, t_3) and k are sent to the adversary $\mathcal{A}$.
8. After receiving the guess b' of the adversary $\mathcal{A}$, output b'.

It is easy to see that if CT^* is an encryption of r_S, then the challenger perfectly simulate the game, and k is a valid session key. Therefore, with non-negligible advantage, the adversary $\mathcal{A}$ outputs $b' = 1$. If CT^* is an encryption of r, then k sent to the adversary is a random key, and the adversary $\mathcal{A}$ outputs $b' = 0$ with non-negligible advantage. Therefore, we have that the challenger wins the IND-CCA game with the same advantage of that of $\mathcal{A}$, i.e.,

$$\mathsf{Adv}^{\mathtt{IND\text{-}CCA}}(\mathcal{C}) = \mathsf{Adv}^{\mathtt{sk}}(\mathcal{A}).$$

□

References

[1] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 26–45, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

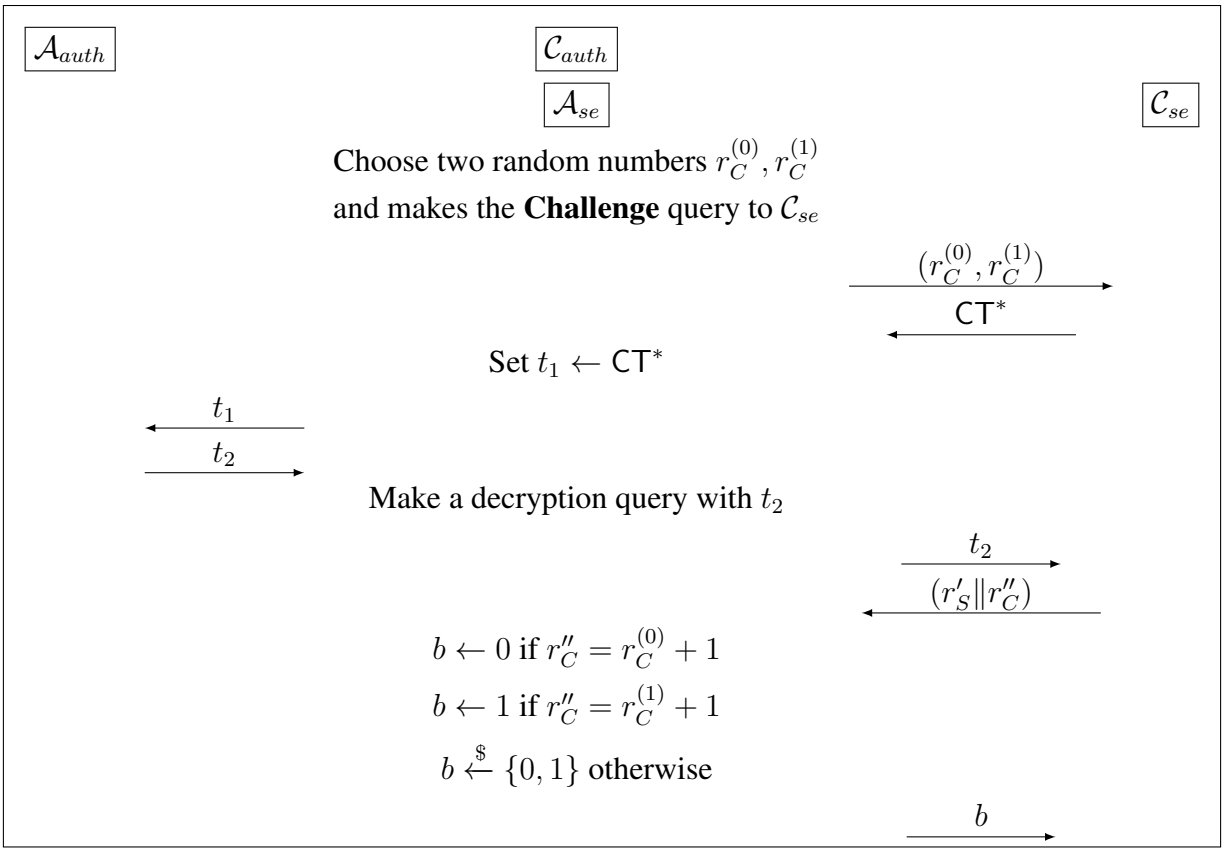

Figure 4: The Authenticity of the Server

[2] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO 93, page 232249, Berlin, Heidelberg, 1993. Springer-Verlag.

[3] M. Bellare and P. Rogaway. Optimal asymmetric encryption – how to encrypt with RSA. pages 92–111. Springer-Verlag, 1995.

[4] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, pages 453–474, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

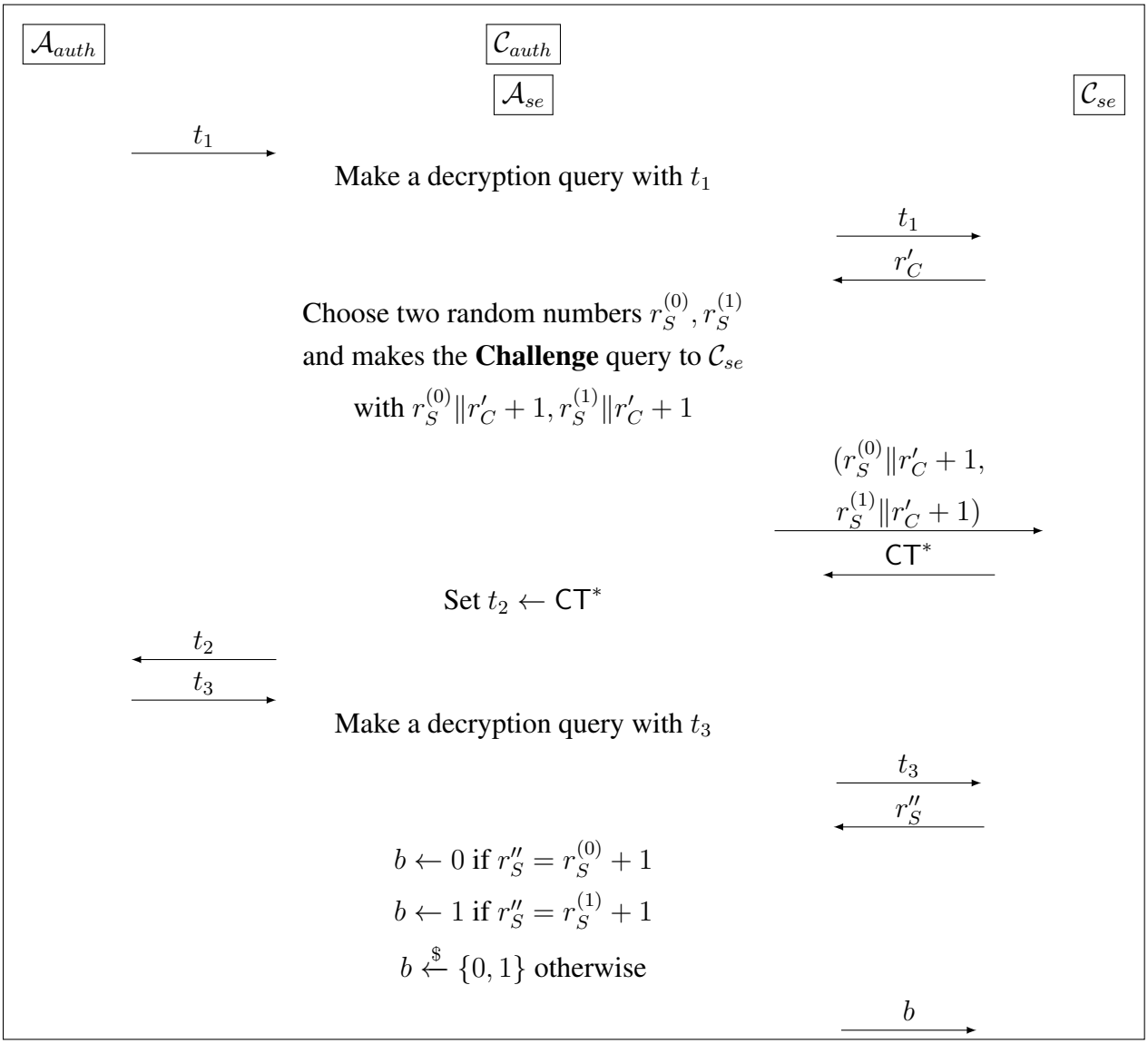

Figure 5: The Authenticity of the Client

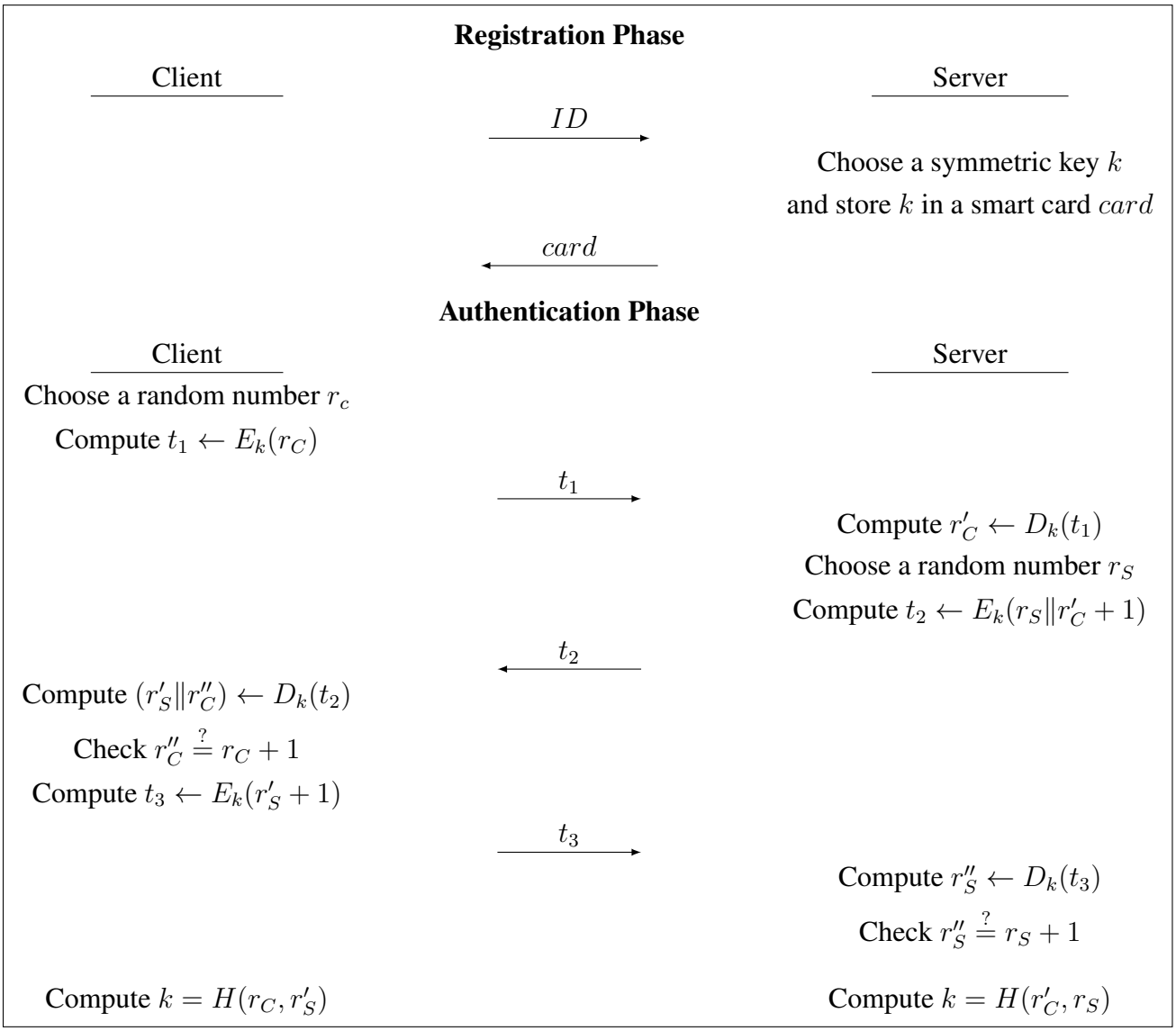

Figure 6: Nonce-Based Authentication