Chapter 6

Key Exchange

Going forward, we shall see that one of the fundamental requirements will be for Alice and Bob to be in possession of a shared secret. In this chapter we shall see how Alice and Bob can do so even if there is an eavesdropper Eve listening in on their conversation.

We shall start with some preliminary definitions and tools before we formally define key agreement, and finally provide a construction building on the tools.

6.1 Groups

We start by defining the notion of a group and its associated properties. A group $\mathbb{G}$ is a non-empty set with an associated binary operation $\bullet$ that takes in two elements, and outputs a third. The binary operator needs to satisfy additional properties, that are listed below.

Definition 25. $(\mathbb{G}, \bullet)$ *is a group if it satisfies the following conditions:*

Closure: *For all* $a, b \in \mathbb{G}$*, we have* $a \bullet b \in \mathbb{G}$.
Associativity: *For all* $a, b, c \in \mathbb{G}$*, we have* $(a \bullet b) \bullet c = a \bullet (b \bullet c)$.
Identity *There exists an identity element* e *such that for all* $a \in \mathbb{G}$*, we have* $e \bullet a = a$.
Inverse *For every* $a \in \mathbb{G}$*, there exists* $b \in \mathbb{G}$ *such that* $a \bullet b = e$.

Note that the above definition doesn't require the group operator to be commutative, i.e. does not require $a \bullet b = b \bullet a$. Try to think of an example of a group for which this is not true. Groups that additionally satisfy the commutative property are called *Abelian Groups*. In this course, we will restrict our attention to abelian groups.

While the above generic definition of a group is useful, we shall find it convenient to consider concrete examples of groups.

Example 15. *A simple example is the group $(\mathbb{Z}, +)$ of integers with the addition operator. It is easy to that it satisfies the properties listed in the definition. One should consider whether the same is true if we replace addition with multiplication, i.e. is $(\mathbb{Z}, \times)$ a group?*

6.1.1 Cyclic Groups

For our course we shall primarily consider groups that are additionally structured, and are referred to as cyclic groups.

Definition 26. *A group $(\mathbb{G}, \cdot)$ is a cyclic group if is generated by a single element, i.e. there exists a $g \in \mathbb{G}$ such that $\mathbb{G} = \{g^0, g^1, \cdots, g^{n-1}\}$ where $g^i = \underbrace{g \cdot g \cdots g}_{i \text{ times}}$. Such a cyclic group $\mathbb{G}$ is denoted as $\mathbb{G} = \langle g \rangle$ and g is called the* generator *of the group.*

From the above definition, it is clear that the order (size) of $\mathbb{G}$ is $|\mathbb{G}| = n$. When clear from context, we shall often drop the group operator to avoid cluttered notation.

6.2 Hard Problems in Cyclic Groups

In this section, we will look at some problems that are believed to be hard for certain cyclic groups. We will then leverage the hardness of one of the problems to construct a key agreement scheme. For each of the problems below we consider a cyclic group $\mathbb{G}$ of order p, where p is an n-bit *safe prime* number (i.e. $p = 2q + 1$ for some large prime q).

6.2.1 Discrete Log (DL)

Since $\mathbb{G}$ is cyclic, every element can be represented uniquely as g^a for some $a \mod p$. It turns out that given an x, it is hard to find the corresponding a such that $g^{a \mod p} = x$. Of course this can't hold true for all x, and we formalize this below requiring the computation of a *discrete log* for a random x. Unless explicitly required, we drop the $\mod p$ from the exponent and assume values in the exponent are always from $\{0, \cdots, p-1\}$.

We say $a \equiv b \mod p$ if there exists an integer k such that $a = kp + b$, i.e. b is the remained when a is divided by p and takes values only in $\{0, \cdots, p-1\}$.

Definition 27 (Discrete Log (DL) Assumption). *Let $\mathbb{G}$ be a cyclic group of order p (where p is a safe prime) with generator g, then for every non-uniform PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(n)$ such that*

$$\Pr[a \leftarrow\$ \{0, \cdots, p-1\}, \widetilde{a} \leftarrow \mathcal{A}(\mathbb{G}, p, g, g^a) : \widetilde{a} = a] \leq \mathsf{negl}(n)$$

The DL *problem* is to find a, while the *assumption* is that the DL problem is hard.

As we've seen throughout, we shall find it convenient to describe the above definition as a game the DL problem in Figure 6.1 between the DL Challenger and adversary $\mathcal{A}_{\mathsf{DL}}$, where the adversary wins if the challenger outputs 1. We require that the probability $\mathcal{A}_{\mathsf{DL}}$ wins is bounded by $\mathsf{negl}(n)$.

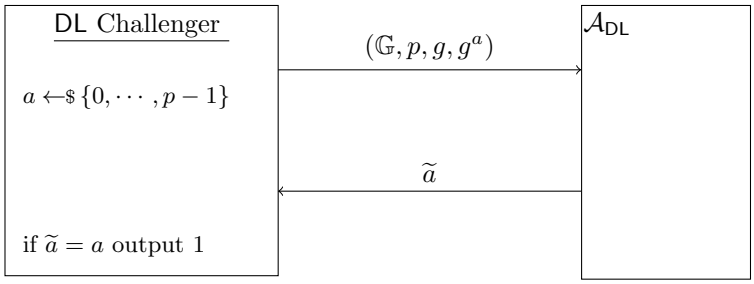

Figure 6.1: DL Challenge Game

6.2.2 Computational Diffie-Hellman (CDH)

It turns out that while DL is a natural problem, we do not know how to leverage it get primitives we want. Instead, we define two other related problem. We start with the first, the *computational Diffie-Hellman* assumption (CDH). Here, an adversary is given g^a and g^b and asked to compute g^{ab}. Intuition would suggest that this is a more difficult problem than DL, and we will indeed formalize this intuition in Section 6.2.4. We formally define the problem below.

Definition 28 (Computational Diffie-Hellman Problem (CDH) Assumption). *Let $\mathbb{G}$ be a cyclic group of order p (where p is a safe prime)*

with generator g, then for every non-uniform PPT *adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(n)$ *such that*

$$\Pr\left[a, b \leftarrow\$ \{0, \cdots, p-1\}, y \leftarrow \mathcal{A}(\mathbb{G}, p, g, g^a, g^b) : y = g^{ab}\right] \leq \mathsf{negl}(n)$$

The corresponding game the CDH problem between the CDH Challenger and adversary $\mathcal{A}_{\mathsf{CDH}}$ is given in Figure 6.2.

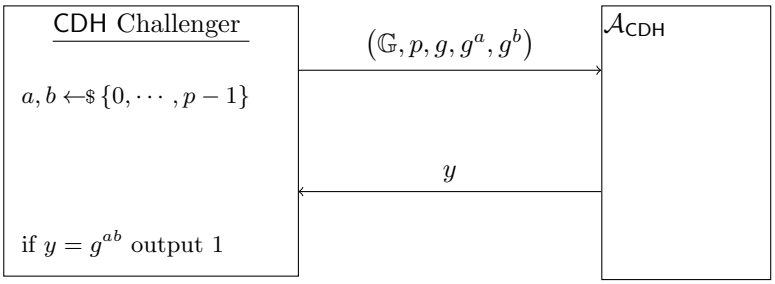

Figure 6.2: CDH Challenge Game

6.2.3 Decisional Diffie-Hellman (DDH)

The above definitions have a flavor similar to one-way functions, where we ask the adversary to compute some value given some information. The next definition is of the indistinguishability flavor, where we simply ask the adversary to distinguish between two distributions. Specifically, we give the adversary either (g^a, g^b, g^{ab}) or (g^a, g^b, g^r), and ask that it distinguish the two for a random r. This is formalized below.

Definition 29 (Decisional Diffie-Hellman (DDH) Assumption). *Let $\mathbb{G}$ be a cyclic group of order p (where p is a safe prime) with generator g, then the following two distributions are computationally indistinguishable:*

- $\left\{a, b \leftarrow\$ \{0, \cdots, p-1\} : \left(\mathbb{G}, p, g, g^a, g^b, g^{ab}\right)\right\}$
- $\left\{a, b, r \leftarrow\$ \{0, \cdots, p-1\} : \left(\mathbb{G}, p, g, g^a, g^b, g^r\right)\right\}$

The corresponding game for the DDH problem between the DDH Challenger and adversary $\mathcal{A}_{\mathsf{DDH}}$ is given in Figure 6.3.

Remark 7. *It should be noted that the* DDH *assumption as stated above is not*

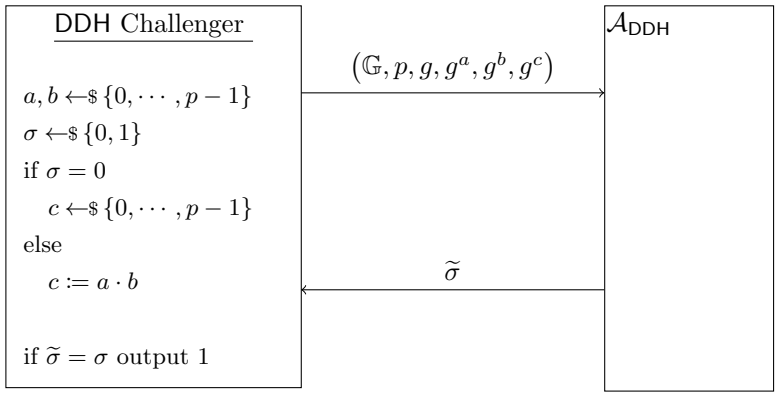

Figure 6.3: DDH Challenge Game

necessarily true in all cyclic groups and care must be taken when we pick the group $\mathbb{G}$ to work with.

6.3 Key Agreement

Now that we have the tools, let us formally define a key agreement (or exchange) protocol.

Definition 30 (Key Agreement Protocol). *A key agreement protocol is an efficient protocol executed by two parties Alice and Bob. During protocol execution, Alice uses randomness r_A while Bob uses randomness r_b. The protocol transcript of the execution is $\tau := \tau(r_A, r_B)$; Alice's output is $k_A := k_A(\tau, r_A)$ and Bob's output is $k_B := k_B(\tau, r_B)$. The protocol satisfies the following properties:*

Correctness

$$\Pr_{r_A, r_B}[k_A = k_B] = 1 - \mathsf{negl}(n)$$

Security

$$(k_A, \tau) \equiv (k_B, \tau) \approx_c (r, \tau)$$

where the eavesdropper Eve's view is simply the transcript τ, and r is a uniformly sampled string independent of τ.

The definition states that even given the transcript, an adversarial Eve who simply eavesdrops cannot tell the difference between the established key k_A (or k_B) from a truly random string independent of τ.

6.4 Construction of Key Agreement Protocol based on DDH

In this section, we finally construct a key agreement protocol based on the hardness of the Decisional Diffie-Hellman problem (Section 6.2.3)[?].

Make sure to map the protocol to the algorithms from Defintion 30 and convince yourself why the security holds. Note that g^r for a random r can be considered to be uniformly random.

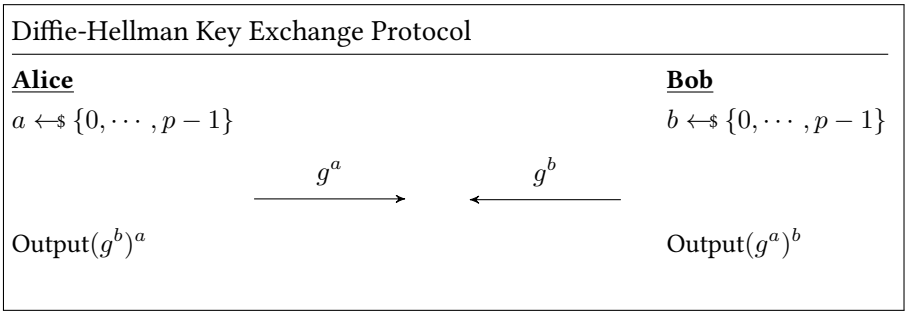

Figure 6.4: Diffie-Hellman Key Exchange Protocol

The security follows directly from the definition of the key agreement protocol and the DDH assumption.

Remark 8. *A few remarks regarding the above protocol are in order.*

1. *Note that we only prove the above protocol secure against eavesdropping adversaries. These are the mildest form of adversaries, since one might expect an adversary to modify the messages sent between Alice and Bob. We call such adversaries active, and one consider whether the above protocol remains secure against active adversaries.*
2. *We needed the stronger assumption of* DDH *since a weaker assumption, say, like* CDH *doesn't prevent half the bits of the shared key from being leaked.*

Chapter 7

Secret-Key Encryption

In this chapter we shall talk about encryption, the most commonly associated term when one thinks of cryptography. Historically, cryptography was used almost exclusively for encryption. Over this and subsequent chapters we shall see many wonderful applications of cryptography using the tools we've discussed so far, but also developing further tools along the way..

7.1 Setting

We assume that Alice and Bob share a secret $s \in \{0,1\}^n$. In Chapter 6 we have seen how Alice and Bob can establish such a shared secret. Alice wants to send a private message to Bob, and we assume the "worst case" that the communication channel is public such that a third party Eve can read all messages sent on the channel. Alice will use an encryption scheme to communicate over the public channel with Bob.

7.2 Secret-key Encryption

We establish first the syntax, then talk about correctness and security of the encryption scheme.

Syntax A secret-key encryption consists of three algorithms described below:

> **Key Generation.** $\mathsf{KGen}(1^n)$ is a randomized algorithm that takes no input, and outputs randomly sampled key s.
>
> **Encryption.** $\mathsf{Enc}(s, m)$ may either be a deterministic or randomized algorithm that takes as input a key s, message m and

outputs a ciphertext c.
When the encryption is randomized, we will sometimes explicitly add an additional input to include randomness as $\mathsf{Enc}(s, m; r)$ where r is the randomness. Note that when the randomness is explicit, Enc is a deterministic algorithm, i.e. randomness has already been accounted for in the input.

Decryption. $\mathsf{Dec}(s, c)$ is a deterministic algorithm that takes as input a key s, ciphertext c and outputs a message m.

Each of these algorithms must run in polynomial time. In the above scenario, to send a message m to Bob, Alice uses the encryption algorithm to compute $c \leftarrow \mathsf{Enc}(s, m)$ and sends c on the public channel to Bob. Bob then uses the decryption algorithm $m := \mathsf{Dec}(s, c)$ to recover the message.
A secret-key encryption scheme must satisfy the two properties described below:

Correctness We require that the scheme satisfies correctness, where decrypting an encrypted message should yield the message.

$$\Pr[s \leftarrow \mathsf{KGen}(1^n), c \leftarrow \mathsf{Enc}(s, m) \ : \ \mathsf{Dec}(s, c) = m] = 1$$

where the probability is over the randomness used in the key generation and encryption algorithms.
For simplicity, since the probability is 1, this can be restated as

$$\forall m, \ \mathsf{Dec}(s, \mathsf{Enc}(s, m)) = m,$$

where $s \leftarrow \mathsf{KGen}(1^n)$.

Security Intuitively an eavesdropper Eve must not be able to decipher the message m from the ciphertext c. More specifically, we require that she cannot distinguish between ciphertexts of two different messages m and m'. To formalize this we introduce the notion of IND-CPA Security (Indistinguishability under Chosen Plaintext Attack). Note here that n is known as the "security parameter" which expresses the degree of security of the scheme.

Definition 31 (IND-CPA). *A secret-key encryption scheme* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* IND-CPA *secure if for all non uniform* PPT *ad-*

versaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{r} s \leftarrow \mathsf{KGen}(1^n), \\ (m_0, m_1) \leftarrow \mathcal{A}(1^n), \\ b \leftarrow\!\$\ \{0,1\} \end{array} : \mathcal{A}(\mathsf{Enc}(s, m_b)) = b \right] \leq \frac{1}{2} + \mathsf{negl}(n)$$

This is a game based definition as you've seen before, but importantly the adversary is allowed to pick the messages m_0 and m_1 of its choice. The game is described in Figure 7.1.

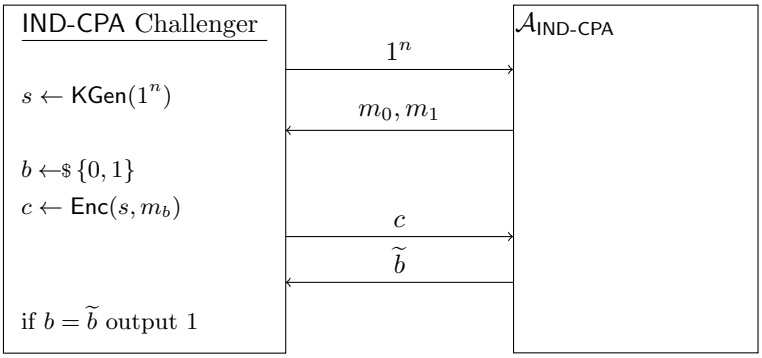

Figure 7.1: IND-CPA security of secret key encryption

Remark 9. *Note the lengths of the messages must be the same to prevent a simple attack that inspects the length of the ciphertext in order to differentiate.*

Often it is easier to think of this definition as requiring computationally indistinguishability of the ciphertexts:

$$\mathsf{Enc}(m_0) \approx_c \mathsf{Enc}(m_1)$$

7.2.1 One-Time Pads

Let's start off constructing a simple scheme satisfying the above definition. Consider the following scheme

$\mathsf{KGen}(1^n)$
1 : $s \leftarrow\!\$\ \{0,1\}^n$
2 : Return s

$\mathsf{Enc}(s, m)$
1 : $c := s \oplus m$
2 : Return c

$\mathsf{Dec}(s, c)$
1 : $m := s \oplus c$
2 : Return m

Correctness follows trivially from the properties of XOR ($\oplus$). For security since s is random, $s \oplus m$ is also random so m is hidden from an informational theoretic perspective. That is to say $\mathsf{Enc}(s, m) \equiv U_n$, which is to say that they are identically distributed. Thus two ciphertexts are more than just indistinguishable, they are in fact identically distributed.

$$\{s \leftarrow \mathsf{KGen}(1^n) : \mathsf{Enc}(s, m_0)\} \equiv \{s \leftarrow \mathsf{KGen}(1^n) : \mathsf{Enc}(s, m_1)\}$$

Think of why even leaking the XOR of two messages is a bad idea.

Unfortunately, if we try to use the same key to encrypt multiple messages, the security no longer holds. This too follows from the properties of XOR, consider two ciphertexts c_1 and c_2 encrypted under the same key s, $c_1 \oplus c_2 = (s \oplus m_1) \oplus (s \oplus m_1) = m_1 \oplus m_2$ which can break the security.

7.2.2 Encryption using PRG

In the simple scheme that we saw above, we noted that a key can never be reused, meaning that the length of the secret-key grows with the length of the message being encrypted. We now discuss an encryption scheme where a secret-key of a fixed length can be used to encrypt a polynomially long message.

We will construct such an encryption using pseudorandom generators (PRG) (Definition 22) by relying on the fact that the output of a PRG is computationally indistinguishable from a uniformly random string. The intuition is that we can use a PRG to convert a random key of a fixed length into a pseudorandom key of the necessary length to encrypt messages that are of arbitrary polynomial length.

Consider the following encryption scheme where the length of the message is $\ell(n)$ and $G : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ is the PRG:

$\mathsf{KGen}(1^n)$	$\mathsf{Enc}(s, m)$	$\mathsf{Dec}(s, c)$
1 : $s \xleftarrow{\$} \{0,1\}^n$	1 : $c := G(s) \oplus m$	1 : $m := G(s) \oplus c$
2 : Return s	2 : Return c	2 : Return m

For security we do not have the identical distribution to uniform random as we did with one-time pads, instead we show security via indistinguishability.

Proposition 2 (Security of Encryption using PRG).

$$\{s \leftarrow \mathsf{KGen}(1^n) : \mathsf{Enc}(s, m_0)\} \approx_c \{s \leftarrow \mathsf{KGen}(1^n) : \mathsf{Enc}(s, m_1)\}$$

Proof Security is proven via a hybrid argument. Rather than using the hybrid lemma though we prove in the forward direction by using the fact that indistinguishability is transitive over polynomial number of hybrids. Consider then

the following list of hybrids, where the change between hybrids have been highlighted:

H_0

1 : $s \leftarrow \mathsf{KGen}(1^n)$
2 : $r := G(s)$
3 : $c := m_0 \oplus r$

H_1

1 : $s \leftarrow \mathsf{KGen}(1^n)$
2 : $r \leftarrow\!\!\$\ \{0,1\}^{\ell(n)}$
3 : $c := m_0 \oplus r$

H_2

1 : $s \leftarrow \mathsf{KGen}(1^n)$
2 : $r \leftarrow\!\!\$\ \{0,1\}^{\ell(n)}$
3 : $c := m_1 \oplus r$

H_3

1 : $s \leftarrow \mathsf{KGen}(1^n)$
2 : $r := G(s)$
3 : $c := m_1 \oplus r$

Note that the output of each hybrid is simply the ciphertext c. From the description of the hybrids, it is clear that $\mathsf{H}_0 = \{s \leftarrow \mathsf{KGen}(1^n) : \mathsf{Enc}(s, m_0)\}$ and $\mathsf{H}_3 = \{s \leftarrow \mathsf{KGen}(1^n) : \mathsf{Enc}(s, m_1)\}$.

First, we show that H_0 and H_1 are computationally indistinguishable by relying on the security of the PRG. For contradiction, let $\mathcal{A}_{\mathsf{H}_0,\mathsf{H}_1}$ be the adversary that distinguishes between H_0 and H_1, using $\mathcal{A}_{\mathsf{H}_0,\mathsf{H}_1}$ we will construct an adversary $\mathcal{A}_{\mathsf{PRG}}$ that we will use to win the PRG game. The reduction is presented in Figure 7.2.

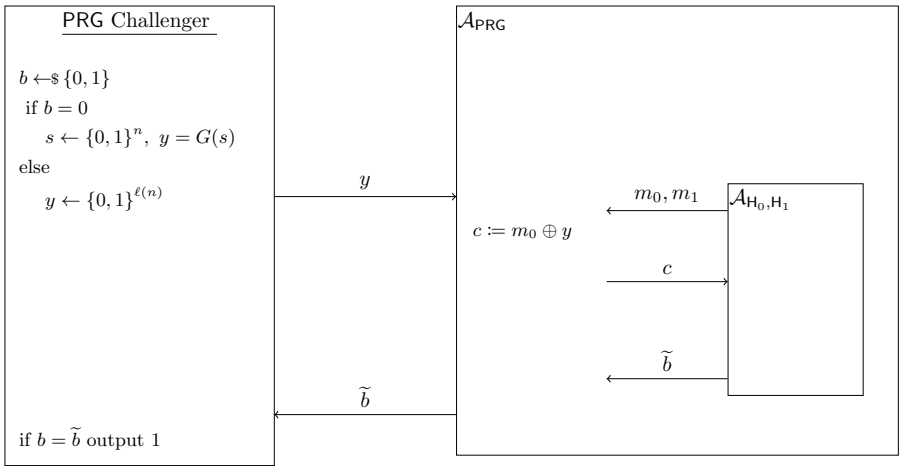

Figure 7.2: Indistinguishability of H_0 and H_1

The next pair of hybrids H_1 and H_2 are (information theoretically) indistin-

guishable by the indistinguishability of one-time pads as we've seen before. Finally H_2 and H_3 are indistinguishable by a symmetric argument to H_0 and H_1. Thus by transitivity of the Hybrid Lemma (Lemma 9), H_0 and H_3 are indistinguishable, which establishes that the scheme is IND-CPA secure. □

Remark 10. *Note that in Hybrid H_1 we are creating a ciphertext that cannot be decrypted correctly since it doesn't follow the encryption procedure. Here correctness does not matter since the adversary does not have access to the decryption procedure to check if the ciphertext is indeed computed correctly. It suffices that the first and last hybrids compute ciphertexts that are* correctly formed.

7.3 Multi-message Secure Encryption

So far, we have only discussed encryption schemes where a key can be used to encrypt a *single* message. In practice we want to be able to encrypt multiple messages. We extend our definition to account for multiple messages.

How does one extend the definition to consider adversaries that may *adaptively* choose the message pairs?

Definition 32 (Multi-message Secure Encryption). *A secret-key encryption scheme* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is multi-message* IND-CPA *secure if for all non-uniform* PPT *adversaries $\mathcal{A}$, for all polynomials $\mathsf{q}(\cdot)$ there exists a negligible function* negl *such that:*

$$\Pr\left[\begin{array}{r} s \xleftarrow{\$} \mathsf{KGen}(1^n), \\ \{(m_0^i, m_1^i)\}_{i=1}^{\mathsf{q}(n)} \leftarrow \mathcal{A}(1^n), \\ b \xleftarrow{\$} \{0,1\} \end{array} : \mathcal{A}\left(\{\mathsf{Enc}(m_b^i)\}_{i=1}^{\mathsf{q}(n)}\right) = b \right] \leq \frac{1}{2} + \mathsf{negl}(n)$$

This definition is very similar to our first definition, but now in the security game the adversary sends two arrays of messages,$(m_0^1, \cdots, m_0^{\mathsf{q}(n)})$ and $(m_1^1, \cdots, m_1^{\mathsf{q}(n)})$ and the challenger encrypts one of the arrays and returns an array of ciphertexst $(c^1, \ldots, c^{\mathsf{q}(n)})$. We let $\mathsf{q}(\cdot)$ be any arbitrary polynomial chosen by the adversary $\mathcal{A}$. The game is shown in Figure 7.3.

Theorem 14 (Stateful Multi-message Encryption). *There exists a multi-message secret-key encryption scheme based on* PRG*s where the encryption algorithm is stateful.*

The proof of the above theorem is straightforward and left as an exercise. Very roughly, the idea is that we can expand the key to a sufficiently long pseudorandom string using a PRG (as in the previous construction) and then use different

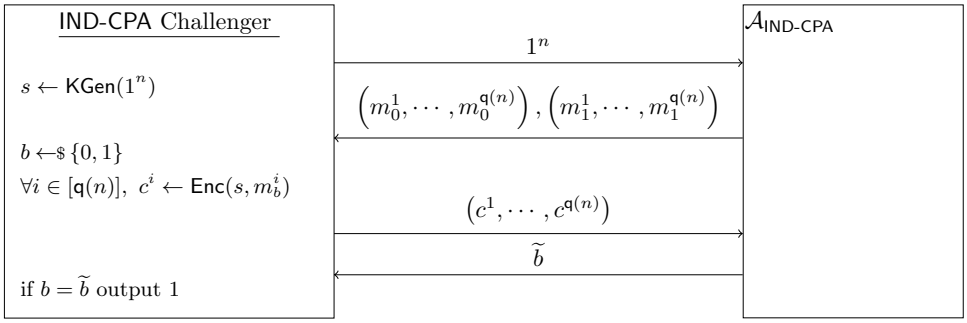

Figure 7.3: Multi-message Secure Encryption

chunks of the PRG output to encrypt different messages. We need to keep track of which chunk is used to encrypt which message, and therefore the encryption algorithm is stateful.

In practice, however, having a stateful encryption algorithm is not very desirable. Instead, we would like to construct multi-message encryption schemes where the encryption algorithm is stateless. The theorem below states that such an encryption scheme must also have a randomized encryption procedure.

Theorem 15 (Randomized Encryption). *A multi-message secure encryption scheme cannot be deterministic and stateless.*

Proof Suppose such a scheme existed. Then an adversary $\mathcal{A}$ could send (m_0^1, m_0^2) and (m_1^1, m_1^2) such that $m_0^1 = m_0^2$ and $m_1^1 \neq m_1^2$. Since no state is kept and the algorithm is entirely deterministic, this gives $\mathsf{Enc}(m_0^1) = \mathsf{Enc}(m_0^2)$ as $m_0^1 = m_0^2$. $\mathcal{A}$ could then just check if $c_1 = c_2$ thus breaking security (as this will be true if $b = 0$ and false with high probability otherwise). □

7.3.1 Encryption using PRFs

To construct a stateless multi-message secure encryption scheme, we will use a family of PRFs (Definition 24). Consider the following encryption scheme where the length of the message is n and $f_s : \{0,1\}^n \mapsto \{0,1\}^n$ is the PRF family:

$\mathsf{KGen}(1^n)$
1 : $s \leftarrow\$ \{0,1\}^n$
2 : Return s

$\mathsf{Enc}(s, m)$
1 : $r \leftarrow\$ \{0,1\}^n$
2 : $c \coloneqq f_s(r) \oplus m$
3 : Return (r, c)

$\mathsf{Dec}(s, (r, c))$
1 : $m \coloneqq f_s(r) \oplus c$
2 : Return m

Theorem 16. *Let* (KGen, Enc, Dec) *be as constructed based on a secure* PRF *family, then it is a multi-message secure encryption scheme.*

Proof The proof is done via a forward hybrid argument. Let F be a truly random function. Consider the following list of hybrids:

H_0

1 : $s \leftarrow \mathsf{KGen}(1^n)$
2 : $\forall i \in [\mathsf{q}(n)]$
3 : $r^i \leftarrow\$ \{0,1\}^n$
4 : $R^i := f_s(r^i)$
5 : $c^i := m_0^i \oplus R^i$

H_1

1 : $s \leftarrow \mathsf{KGen}(1^n)$
2 : $\forall i \in [\mathsf{q}(n)]$
3 : $r^i \leftarrow\$ \{0,1\}^n$
4 : $R^i := F(r^i)$
5 : $c^i := m_0^i \oplus R^i$

H_2

1 : $s \leftarrow \mathsf{KGen}(1^n)$
2 : $\forall i \in [\mathsf{q}(n)]$
3 : $r^i \leftarrow\$ \{0,1\}^n$
4 : $R^i \leftarrow\$ \{0,1\}^n$
5 : $c^i := m_0^i \oplus R^i$

H_3

1 : $s \leftarrow \mathsf{KGen}(1^n)$
2 : $\forall i \in [\mathsf{q}(n)]$
3 : $r^i \leftarrow\$ \{0,1\}^n$
4 : $R^i \leftarrow\$ \{0,1\}^n$
5 : $c^i := m_1^i \oplus R^i$

H_4

1 : $s \leftarrow \mathsf{KGen}(1^n)$
2 : $\forall i \in [\mathsf{q}(n)]$
3 : $r^i \leftarrow\$ \{0,1\}^n$
4 : $R^i := F(r^i)$
5 : $c^i := m_1^i \oplus R^i$

H_5

1 : $s \leftarrow \mathsf{KGen}(1^n)$
2 : $\forall i \in [\mathsf{q}(n)]$
3 : $r^i \leftarrow\$ \{0,1\}^n$
4 : $R^i := f_s(r^i)$
5 : $c^i := m_1^i \oplus R^i$

The output of each hybrid is the array $((r^1, c^1), \cdots, (r^{\mathsf{q}(n)}, c^{\mathsf{q}(n)}))$. From the description of the hybrids, it is clear that H_0 corresponds to the encryption of the array $(m_0^1, \cdots, m_0^{\mathsf{q}(n)})$ while H_5 corresponds to the encryption of the array $(m_1^1, \cdots, m_1^{\mathsf{q}(n)})$.

Sketch out the full proof to make sure you understand it.

The proof follows in a similar manner to that of the one-message security of PRG. We sketch in Figure 7.4 how one proves that hybrids H_0 and H_1 are computationally indistinguishable by relying on the security of the PRF. Specifically we use an adversary $\mathcal{A}_{\mathsf{H}_0,\mathsf{H}_1}$ that distinguishes between H_0 and H_1 to construct an adversary $\mathcal{A}_{\mathsf{PRF}}$ that breaks the security of the PRF.

□

7.4 Semantic Security

We look at the final notion of security for encryption, *semantic security* that we define below.

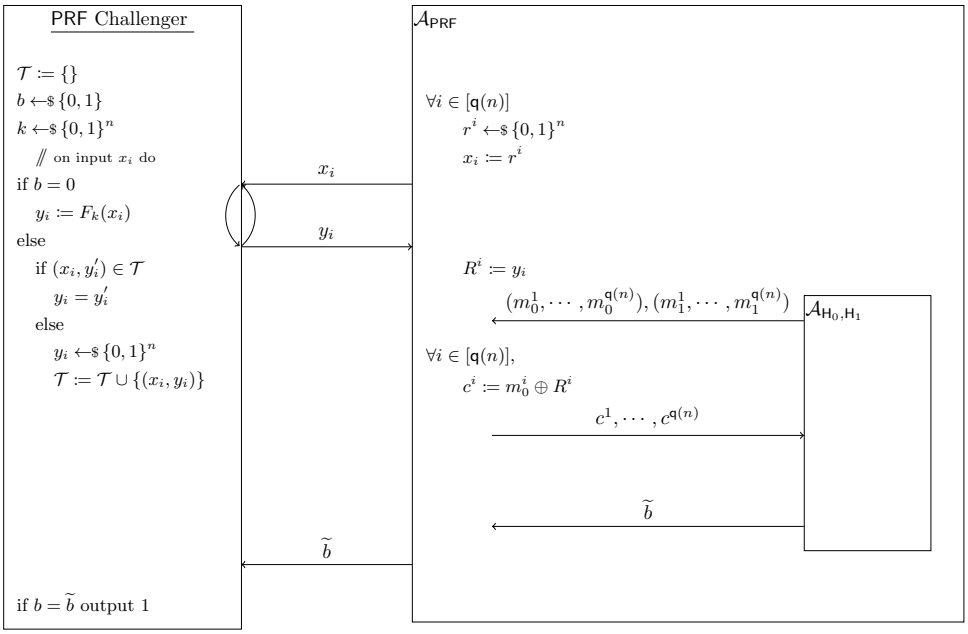

Figure 7.4: Indistinguishability of H_0 and H_1

Definition 33 (Semantic Security). *A secret-key encryption scheme* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* semantically secure *if there exists a* PPT *simulator algorithm* Sim *s.t. the following two experiments generate computationally indistinguishable outputs:*

$$\left\{ \begin{array}{r} (m, z) \leftarrow M(1^n), \\ s \leftarrow \mathsf{KGen}(1^n), \\ (\mathsf{Enc}(s, m), z) \end{array} \right\} \approx_c \left\{ \begin{array}{r} (m, z) \leftarrow M(1^n), \\ \mathsf{Sim}(1^n, z) \end{array} \right\}$$

where M is a machine that randomly samples a message from the message space and arbitrary auxiliary information z.

Intuitively, semantic security promises that a PPT adversary does not learn any "new" information about a message m by simply looking at its ciphertext. To capture the fact that it might already have some information about the underlying message, we denote by z any arbitrary auxiliary information. This intuition is captured by the fact that the adversary's view (ciphertext and auxiliary information z) can be efficiently simulated by Sim given only z and no other information on m.

We now show that semantic security and IND-CPA security are, in fact, equivalent security notions.

Theorem 17. *Semantic security and* IND-CPA*security are equivalent.*

Proof We prove each case separately:

Semantic security $\Rightarrow$ IND-CPA : From semantic security, we have that for any m_1, $(\mathsf{Enc}(m_0), z) \approx_c \mathsf{Sim}(1^n, z)$. Also from semantic security, for any m_1, we have that $(\mathsf{Enc}(m_1), z) \approx_c \mathsf{Sim}(1^n, z)$. Thus, by transitivity, we get that semantic security implies IND-CPA security.

IND-CPA $\Rightarrow$ **semantic security** : From IND-CPA security, we have that encryptions of any two messages are indistinguishable. Then, we simply construct a simulator that encrypts the all zeros string, i.e., $\mathsf{Sim}(1^n, z) = (\mathsf{Enc}(0^n), z)$. Semantic security then immediately follows from IND-CPA security.

□

Chapter 8

Public-Key Encryption

Recall that in the setting of private-key encryption, Alice and Bob were allowed to share a secret before communication. In the public-key setting, Alice and Bob don't share a secret any more. Our goal is to be able to have Alice still send a message to Bob in such a manner that no eavesdropper can distinguish between encryptions of m and m'. We formalize the notion of public-key encryption that allows us to do this.

The syntax is similar to the secret-key encryption, with a few crucial differences.

Syntax A secret-key encryption consists of three algorithms described below:

> **Key Generation.** $\mathsf{KGen}(1^n)$ is a randomized algorithm that takes no input, and outputs a sampled (public-key,secret-key) pair pk, sk.
>
> **Encryption.** $\mathsf{Enc}(\mathsf{pk}, m)$ is a randomized algorithm takes as input a public-key pk, message m and outputs a ciphertext c.
>
> **Decryption.** $\mathsf{Dec}(\mathsf{sk}, c)$ is a deterministic algorithm that takes as input a secret-key sk, ciphertext c and outputs a message m or a special symbol $\perp$ indicating failure to decrypt.

After looking at the security definition, consider if Enc can be a determinstic algorithm in the public-key setting.

As before, we require that all the algorithms run in polynomial time.
A public-key encryption scheme must satisfy the two properties described below:

Correctness We require that the scheme satisfies correctness, where decrypting an encrypted message should yield the message.

$$\forall m,\ \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m,$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^n)$.

Security Again, we want to the encryption scheme to have security against an eavesdropper Eve. We now provide our definition of security. We start with a weak definition (also called selective security).

Definition 34. *A public key encryption scheme is weakly-indistinguishably secure under chosen plaintext attack (*IND-CPA*) if for all non uniform* PPT *adversaries $\mathcal{A}$ there exists a negligible function* negl *such that*

$$\Pr\left[\begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KGen}(1^n),\\ (m_0, m_1) \leftarrow \mathcal{A}(1^n), \quad : \mathcal{A}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_b)) = b\\ \qquad\qquad b \leftarrow\!\!\$\ \{0,1\}\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(n)$$

This definition is very similar to the IND-CPA definition (Definition 31) for secret-key encryption. The primary difference being that that $\mathcal{A}$ is additionally given the public-key pk along with the ciphertext. The corresponding game is provided in Figure 8.1.

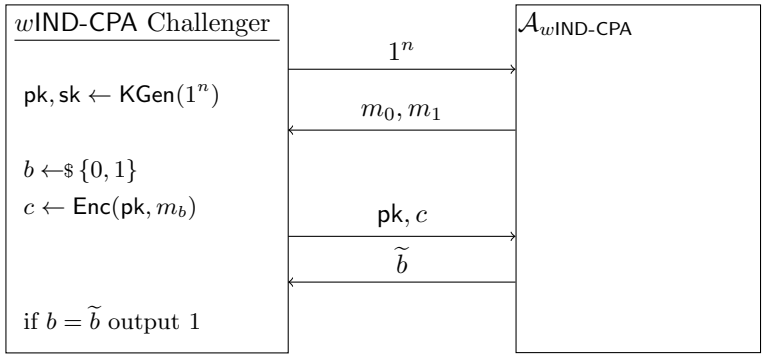

Figure 8.1: Weakly-indistinguishably PKE scheme under chosen plaintext attack (IND-CPA).

We think of this as a weaker notion since the adversary has to choose its message pair prior to looking at the public-key, which as the name suggests is public. We now adapt the above definition to allow the adversary to choose its messages adaptively based on the public-key. send us messages.

Definition 35. *A public key encryption scheme is indistinguishably secure under chosen plaintext attack (*IND-CPA*) if for all non uniform* PPT *adver-*

saries $\mathcal{A}$ there exists a negligible function negl *such that*

$$\Pr\left[\begin{array}{r} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KGen}(1^n), \\ (m_0,m_1) \leftarrow \mathcal{A}(1^n,\mathsf{pk}), \\ b \leftarrow\!\!\$\ \{0,1\} \end{array} : \mathcal{A}(\mathsf{pk},\mathsf{Enc}(\mathsf{pk},m_b)) = b \right] \leq \frac{1}{2}+\mathsf{negl}(n)$$

The corresponding game is provided in Figure 8.2.

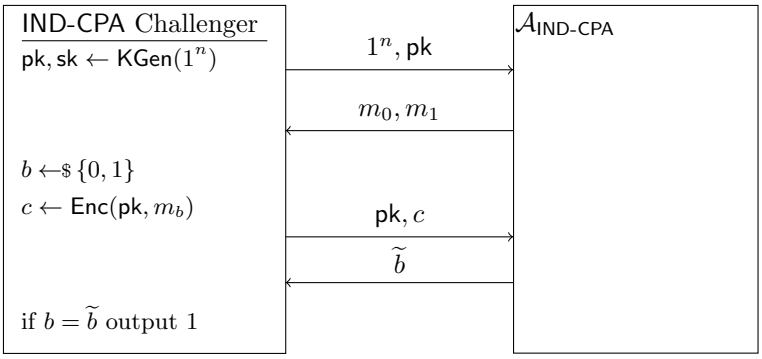

Figure 8.2: Indistinguishably PKE scheme under chosen plaintext attack (IND-CPA).

Just as in the secret key encryption schemes we want to obtain multiple message security as well. However examining our definition we can realize that one-message security already implies multi-message security. Multi-message security is defined analogously to the secret-key encryption setting, and we omit the definition here.

Lemma 18. *One-message security implies multi-message security for public-key encryption*

Proof We now briefly sketch the proof. For simplicity, we only show that one-message security implies two-message security. The general case can be derived in a similar manner.

Suppose that (m_0^1, m_1^1) is the first message pair and (m_0^2, m_1^2) is the second message pair. Now, consider the following sequence of hybrid experiments:

H_0	H_1	H_2
1 : $c^1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0^1)$	1 : $c^1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1^1)$	1 : $c^1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1^1)$
2 : $c^1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0^2)$	2 : $c^1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0^2)$	2 : $c^1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1^2)$

H_0 corresponds to the case in the multi-message game when $b = 0$, and H_2 corresponds to the case that $b = 1$. We start by showing that hybrids H_0 and H_1 are indistinguishable from the one-message security of the public-key encryption scheme. Specifically, if there is an adversary $\mathcal{A}_{\mathsf{H}_0,\mathsf{H}_1}$ that distinguishes between hybrids H_0 and H_1, we use $\mathcal{A}_{\mathsf{H}_0,\mathsf{H}_1}$ to construct an adversary $\mathcal{A}_{1\text{-msg-IND-CPA}}$ that breaks the one-message security of the encryption scheme. The reduction is presented in Figure 8.3.

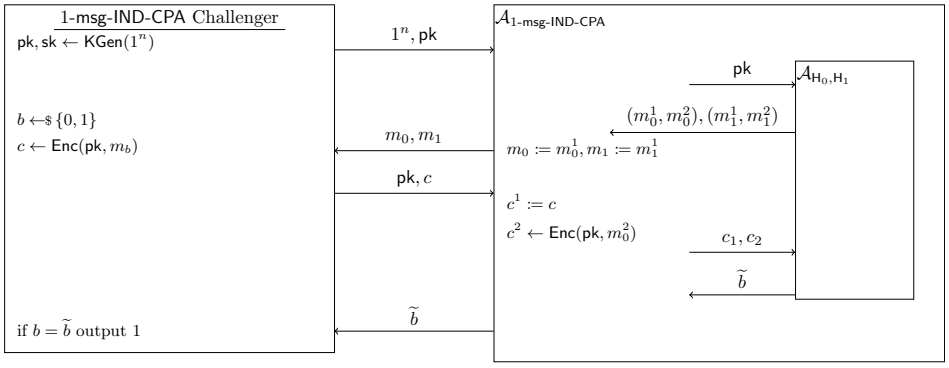

Figure 8.3: Indistinguishability of hybrids H_0 and H_1.

Think of why this approach doesn't work in the secret-key setting. Can you compute the other ciphertexts in the hybrids there?

In the reduction note that if c is an encryption of $m_0 = m_0^1$, then the above corresponds to H_0, otherwise, it corresponds to H_1. Therefore, if $\mathcal{A}_{\mathsf{H}_0,\mathsf{H}_1}$ distinguishes with noticeable probability, then so does$\mathcal{A}_{1\text{-msg-IND-CPA}}$. This contradicts the one-message security of the encryption scheme.

The indistinguishability of hybrids H_1 and H_2 follow in an identical manner. Therefore, by the Hybrid Lemma (Lemma 9) we get that H_0 and H_2 are indistinguishable, as desired.

□