Chapter 23

Public Key Encryption Based on Discrete Logarithms

Historically, encryption has been considered the most important part of cryptography. So it is not surprising that there is a vast literature about public key encryption. It is important to note that, in practice, public key encryption is not usually used to encrypt documents. Instead, one uses public key encryption to securely send keys, and the data is encrypted using symmetric encryption.

It is beyond the scope of this book to discuss all known results on public key encryption, or even to sketch all known approaches to designing public key encryption schemes. The goal of this chapter is very modest. We simply aim to give some definitions and to provide two efficient encryption schemes (one secure in the random oracle model and one secure in the standard model). The encryption schemes in this chapter are all based on Elgamal encryption, the "textbook" version of which has already been discussed in Sections 20.3 and 20.4.

Finally, we emphasise that this Chapter only discusses confidentiality and not simultaneous confidentiality and authentication. The reader is warned that naively combining signatures and encryption does not necessarily provide the expected security (see, for example, the discussion in Section 1.2.3 of Joux [317]).

23.1 CCA Secure Elgamal Encryption

Recall that security notions for public key encryption were given in Section 1.3.1. As we have seen, the textbook Elgamal encryption scheme does not have OWE-CCA security, since one can easily construct a related ciphertext whose decryption yields the original message. A standard way to prevent such attacks is to add a message authentication code (MAC); see Section 3.3.

We have also seen (see Section 20.3) that Elgamal can be viewed as static Diffie-Hellman key exchange followed by a specific symmetric encryption. Hence, it is natural to generalise Elgamal encryption so that it works with any symmetric encryption scheme. The scheme we present in this section is known as **DHIES** and, when implemented with elliptic curves, is called **ECIES**. We refer to Abdalla, Bellare and Rogaway [1] or Chapter III of [65] for background and discussion.

Let κ be a security parameter. The scheme requires a symmetric encryption scheme, a MAC scheme and a key derivation function. The symmetric encryption functions Enc and Dec take an l_1-bit key and encrypt messages of arbitrary length. The MAC function MAC takes an l_2-bit key and a message of arbitrary length and outputs an l_3-bit binary string. The key derivation function is a function $\mathsf{kdf} : G \to \{0,1\}^{l_1+l_2}$. The values l_1, l_2 and l_3 depend on the security parameter. Note that it is important that the MAC is evaluated on the ciphertext not the message, since a MAC is not required to have any confidentiality properties. The DHIES encryption scheme is given in Figure 23.1.

KeyGen(κ): Generate an algebraic group or algebraic group quotient G whose order is divisible by a large prime r (so that the discrete logarithm problem in the subgroup of prime order r requires at least 2^κ bit operations).
Choose a random $g \in G$ of exact order r. Choose a random integer $0 < a < r$ and set $h = g^a$.
The public key is (G, g, h) and the private key is a. Alternatively, (G, g) are system parameters that are fixed for all users and only h is the public key.

The message space is $\mathsf{M}_\kappa = \{0,1\}^*$.
The ciphertext space is $\mathsf{C}_\kappa = G \times \{0,1\}^* \times \{0,1\}^{l_3}$.

Encrypt(m, h): ($\mathsf{m} \in \{0,1\}^*$ and h is the authentic public key of the receiver)

- Choose a random $0 < k < r$ and set $\mathsf{c}_1 = g^k$.
- Set $K = \mathsf{kdf}(h^k)$ and parse K as $K_1 \| K_2$ where K_1 and K_2 are l_1 and l_2 bit binary strings respectively.
- Set $\mathsf{c}_2 = \mathsf{Enc}_{K_1}(m)$ and $\mathsf{c}_3 = \mathsf{MAC}_{K_2}(\mathsf{c}_2)$.
- Transmit the ciphertext $(\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3)$.

Decrypt$(\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, a)$:

- Check that $\mathsf{c}_1 \in G$ and that c_3 is an l_3-bit string (if not then return $\perp$ and halt).
- Compute $K = \mathsf{kdf}(\mathsf{c}_1^a)$ and parse it as $K_1 \| K_2$.
- Check whether $\mathsf{c}_3 = \mathsf{MAC}_{K_2}(\mathsf{c}_2)$ (if not then return $\perp$ and halt).
- Return $\mathsf{m} = \mathsf{Dec}_{K_1}(\mathsf{c}_2)$.

Figure 23.1: DHIES Public Key Encryption.

Exercise 23.1.1. Show that decryption does return the message when given a ciphertext produced by the DHIES encryption algorithm.

A variant of DHIES is to compute the key derivation function on the pair of group (or algebraic group quotient) elements (g^k, h^k) rather than just h^k. This case is presented in

Section 10 of Cramer and Shoup [161]. As explained in Section 10.7 of [161], this variant can yield a tighter security reduction.

23.1.1 The KEM/DEM Paradigm

Shoup introduced a formalism for public key encryption that has proved to be useful. The idea is to separate the "public key" part of the system (i.e., the value c_1 in Figure 23.1) from the "symmetric" part (i.e., $(\mathsf{c}_2, \mathsf{c}_3)$ in Figure 23.1). A **key encapsulation mechanism** (or **KEM**) outputs a public key encryption of a random symmetric key (this functionality is very similar to **key transport**; the difference being that a KEM generates a fresh random key as part of the algorithm). A **data encapsulation mechanism** (or **DEM**) is the symmetric part. The name **hybrid encryption** is used to describe an encryption scheme obtained by combining a KEM with a DEM.

More formally, a KEM is a triple of three algorithms (KeyGen, Encrypt and Decrypt)[1] that depend on a security parameter κ. Instead of a message space, there is space $\mathcal{K}_\kappa$ of possible keys to be encapsulated. The randomised algorithm Encrypt takes as input a public key and outputs a ciphertext c and a symmetric key $K \in \mathcal{K}_\kappa$ (where κ is the security parameter). One says that c **encapsulates** K. The Decrypt algorithm for a KEM takes as input a ciphertext c and the private key and outputs a symmetric key K (or $\perp$ if the decryption fails). The Encrypt algorithm for a DEM takes as input a message and a symmetric key K and outputs a ciphertext. The Decrypt algorithm of a DEM takes a ciphertext and a symmetric key K and outputs either $\perp$ or a message.

The simplest way to obtain a KEM from Elgamal is given in Figure 23.2. The DEM corresponding to the hybrid encryption scheme in Section 23.1 takes as input m and K, parses K as $K_1 \| K_2$, computes $\mathsf{c}_2 = \mathsf{Enc}_{K_1}(\mathsf{m})$ and $\mathsf{c}_3 = \mathsf{MAC}_{K_2}(\mathsf{c}_2)$ and outputs $(\mathsf{c}_2, \mathsf{c}_3)$.

KeyGen(κ): This is the same as standard Elgamal; see Figure 23.1.
Encrypt(h): Choose random $0 \leq k < r$ and set $\mathsf{c} = g^k$ and $K = \mathsf{kdf}(h^k)$. Return the ciphertext c and the key K.
Decrypt(c, a): Return $\perp$ if $\mathsf{c} \notin \langle g \rangle$. Otherwise return $\mathsf{kdf}(\mathsf{c}^a)$.

Figure 23.2: Elgamal KEM.

Shoup has defined an analogue of IND-CCA security for a KEM. We refer to Section 7 of Cramer and Shoup [161] for precise definitions for KEMs, DEMs and their security properties, but give an informal statement now.

Definition 23.1.2. An IND-CCA adversary for a KEM is an algorithm A that plays the following game: The input to A is a public key. The algorithm A can also query a decryption oracle that will provide decryptions of any ciphertext of its choosing. At some point the adversary requests a challenge, which is a KEM ciphertext c^* together with a key $K^* \in \mathcal{K}_\kappa$. The challenger chooses K^* to be either the key corresponding to the ciphertext c^* or an independently chosen random element of $\mathcal{K}_\kappa$ (both cases with probability 1/2). The game continues with the adversary able to query the decryption oracle with any ciphertext $\mathsf{c} \neq \mathsf{c}^*$. Finally, the adversary outputs a guess for whether K^* is the key corresponding to c^*, or a random key (this is the same as the "real or random" security notion for key exchange in Section 20.5). Denote by $\Pr(A)$ the success probability of A in this game and define the **advantage** $\mathrm{Adv}(A) = |\Pr(A) - 1/2|$. The KEM is IND-CCA secure if every polynomial-time adversary has negligible advantage.

[1] Sometimes the names Encap and Decap are used instead of Encrypt and Decrypt.

Theorem 5 of Section 7.3 of [161] shows that, if a KEM satisfies IND-CCA security and if a DEM satisfies an analogous security property, then the corresponding hybrid encryption scheme has IND-CCA security. Due to lack of space we do not present the details.

23.1.2 Proof of Security in the Random Oracle Model

We now sketch a proof that the Elgamal KEM of Figure 23.2 has IND-CCA security. The proof is in the random oracle model. The result requires a strong assumption (namely, the Strong-Diffie-Hellman or gap-Diffie-Hellman assumption). Do not be misled by the use of the word "strong"! This computational problem is not harder than the Diffie-Hellman problem. Instead, the assumption that this problem is hard is a *stronger* (i.e., less likely to be true) assumption than the assumption that the Diffie-Hellman problem is hard.

Definition 23.1.3. Let G be a group of prime order r. The **strong Diffie-Hellman problem** (**Strong-DH**) is: Given $g, g^a, g^b \in G$ (where $1 \leq a, b < r$), together with a decision static Diffie-Hellman oracle (**DStatic-DH oracle**) $A_{g,g^a}(h_1, h_2)$ (i.e., $A_{g,g^a}(h_1, h_2) = 1$ if and only if $h_2 = h_1^a$), to compute g^{ab}.

An instance generator for Strong-DH takes as input a security parameter κ, outputs a group G and an element g of prime order r (with $r > 2^{2\kappa}$) and elements $g^a, g^b \in G$ where $1 \leq a, b < r$ are chosen uniformly at random. As usual, we say that Strong-DH is hard for the instance generator if all polynomial-time algorithms to solve the problem have negligible success probability. The **strong Diffie-Hellman assumption** is that there is an instance generator for which the Strong-DH problem is hard.

It may seem artificial to include access to a decision oracle as part of the assumption. Indeed, it is a significant drawback of the encryption scheme that such an assumption is needed for the security. Nevertheless, the problem is well-defined and seems to be hard in groups for which the DLP is hard. A related problem is the **gap Diffie-Hellman problem**: again the goal is to compute g^{ab} given (g, g^a, g^b), but this time one is given a full DDH oracle. In some situations (for example, when using supersingular elliptic or hyperelliptic curves) one can use pairings to provide a DDH oracle and the artificial nature of the assumption disappears. The proof of Theorem 23.1.4 does not require a full DDH oracle and so it is traditional to only make the Strong-DH assumption.

Theorem 23.1.4. *The Elgamal KEM of Figure 23.2, with the key derivation function replaced by a random oracle, is IND-CCA secure if the strong Diffie-Hellman problem is hard.*

Proof: (Sketch) Let (g, g^a, g^b) be the Strong-DH instance and let A_{g,g^a} be the DStatic-DH oracle. Let B be an IND-CCA adversary against the KEM. We want to use B to solve our Strong-DH instance. To do this we will simulate the game that B is designed to play. The simulation starts B by giving it the public key (g, g^a). Note that the simulator does not know the corresponding private key.

Since the key derivation function is now a random oracle, it is necessary for B to query the simulator whenever it wants to compute kdf; this fact is crucial for the proof. Indeed, the whole idea of the proof is that when B requests the challenge ciphertext we reply with $\mathsf{c}^* = g^b$ and with a randomly chosen $K^* \in \mathcal{K}_\kappa$. Since kdf is a random oracle, the adversary can have no information about whether or not c^* encapsulates K^* unless the query $\mathsf{kdf}((\mathsf{c}^*)^a)$ is made. Finally, note that $(\mathsf{c}^*)^a = g^{ab}$ is precisely the value the simulator wants to find.

More precisely, let E be the event (on the probability space of Strong-DH instances and random choices made by B) that B queries kdf on $(\mathsf{c}^*)^a = g^{ab}$. The advantage of B is

$\mathrm{Adv}(B) = \left|\Pr(B) - \frac{1}{2}\right|$ where $\Pr(B)$ is the probability that B wins the IND-CCA security game. Note that $\Pr(B) = \Pr(B|E)\Pr(E) + \Pr(B|\neg E)\Pr(\neg E)$. When kdf is a random oracle we have $\Pr(B|\neg E) = 1/2$. Writing $\Pr(B|E) = 1/2 + u$ for some $-1/2 \leq u \leq 1/2$ we have $\Pr(B) = 1/2 + u\Pr(E)$ and so $\mathrm{Adv}(B) = |u|\Pr(E)$. Since $\mathrm{Adv}(B)$ is assumed to be non-negligible it follows that $\Pr(E)$ is non-negligible. In other words, a successful adversary makes an oracle query on the value g^{ab} with non-negligible probability.

To complete the proof it is necessary to explain how to simulate kdf and Decrypt queries, and to analyse the probabilities. The simulator maintains a list of all queries to kdf. The list is initially empty. Every time that $\mathsf{kdf}(u)$ is queried the simulator first checks whether $u \in G$ and returns $\perp$ if not. Then the simulator checks whether an entry (u, K) appears in the list of queries and, if so, returns K. If no entry appears in the list then use the oracle A_{g,g^a} to determine whether $u = g^{ab}$ (i.e., if $A_{g,g^a}(g^b, u) = 1$). If this is the case then g^{ab} has been computed and the simulation outputs that value and halts. In all other cases, a value K is chosen uniformly and independently at random from $\mathcal{K}_\kappa$, (u, K) is added to the list of kdf queries, and K is returned to B.

Similarly, when a decryption query on ciphertext c is made then one checks, for each pair (u, K) in the list of kdf values, whether $A_{g,g^a}(\mathsf{c}, u) = 1$. If this is the case then return K. If there is no such triple then return a random $K' \in \mathcal{K}$.

One can check that the simulation is sound (in the sense that Decrypt does perform the reverse of Encrypt) and that the outputs of kdf are indistinguishable from random. Determining the advantage of the simulator in solving the strong-DH problem is then straightforward. We refer to Section 10.4 of Cramer and Shoup [161] for a careful proof using the "game hopping" methodology (actually, that proof applies to the variant in Exercise 23.1.5, but it is easily adapted to the general case). □

Exercise 23.1.5. A variant of the scheme has the key derivation function applied to the pair (g^k, h^k) in Encrypt instead of just h^k (respectively, $(\mathsf{c}_1, \mathsf{c}_1^a)$ in Decrypt). Adapt the security proof to this case. What impact does this have on the running time of the simulator?

The IND-CPA security of the Elgamal KEM can be proved in the standard model (the proof is analogous to the proof of Theorem 20.4.10) under the assumption of Definition 23.1.6. The IND-CCA security can also be proved in the standard model under an interactive assumption called the oracle Diffie-Hellman assumption. We refer to Abdalla, Bellare and Rogaway [1] for the details of both these results.

Definition 23.1.6. Let G be a group and $\mathsf{kdf} : G \to \mathcal{K}$ a key derivation function. The **hash Diffie-Hellman problem** (**Hash-DH**) is to distinguish the distributions $(g, g^a, g^b, \mathsf{kdf}(g^{ab}))$ and (g, g^a, g^b, K) where K is chosen uniformly from $\mathcal{K}$. The **hash Diffie-Hellman assumption** is that there exist instance generators such that all polynomial-time algorithms for Hash-DH have negligible advantage.

Exercise 23.1.7. Let G be a group of prime order r and let $\mathsf{kdf} : G \to \{0,1\}^l$ be a key derivation function such that $\log_2(r)/2 < l < \log_2(r)$ and such that the output distribution is statistically close to uniform. Show that DDH $\leq_R$ Hash-DH $\leq_R$ CDH.

An elegant variant of Elgamal, with IND-CCA security in the random oracle model depending only on CDH rather than strong Diffie-Hellman, is given by Cash, Kiltz and Shoup [120].

23.2 Cramer-Shoup Encryption

In their landmark paper [159], Cramer and Shoup gave an encryption scheme with a proof of CCA security in the standard model. Due to lack of space we will only be able to give a sketch of the security analysis of the scheme.

To motivate how they achieve their result, consider the proof of security for the Elgamal KEM (Theorem 23.1.4). The simulator uses the adversary to solve an instance of the CDH problem. To do this one puts part of the CDH instance in the public key (and hence, one does not know the private key) and part in the challenge ciphertext. To prove CCA security we must be able to answer decryption queries without knowing the private key. In the proof of Theorem 23.1.4 this requires a DDH oracle (to determine correct ciphertexts from incorrect ones) and also the use of the random oracle model (to be able to "see" some internal operations of the adversary).

In the random oracle model one generally expects to be able to prove security under an assumption of similar flavour to CDH (see Theorem 20.4.11 and Theorem 23.1.4). On the other hand, in the standard model one only expects[2] to be able to prove security under a decisional assumption like DDH (see Theorem 20.4.10). The insight of Cramer and Shoup is to design a scheme whose security depends on DDH and is such that the entire DDH instance can be incorporated into the challenge ciphertext. The crucial consequence is that the simulator can now generate public and private keys for the scheme, run the adversary, and be able to handle decryption queries.

The proof of security hinges (among other things) on the following result.

Lemma 23.2.1. *Let G be a group of prime order r. Let $g_1, g_2, u_1, u_2, h \in G$ with $(g_1, g_2) \neq (1, 1)$. Consider the set*

$$\mathcal{X}_{g_1,g_2,h} = \{(z_1, z_2) \in (\mathbb{Z}/r\mathbb{Z})^2 \ : \ h = g_1^{z_1} g_2^{z_2}\}.$$

Then $\#\mathcal{X}_{g_1,g_2,h} = r$. Let $0 \leq k < r$ be such that $u_1 = g_1^k$. If $u_2 = g_2^k$ then $u_1^{z_1} u_2^{z_2} = h^k$ for all $(z_1, z_2) \in \mathcal{X}_{g_1,g_2,h}$. If $u_2 \neq g_2^k$ then

$$\{u_1^{z_1} u_2^{z_2} : (z_1, z_2) \in \mathcal{X}_{g_1,g_2,h}\} = G.$$

Exercise 23.2.2. Prove Lemma 23.2.1.

Figure 23.3 presents the basic **Cramer-Shoup encryption scheme**. The scheme requires a group G of prime order r and the message m is assumed to be an element of G. Of course, it is not necessary to "encode" data into group elements, in practice one would use the Cramer-Shoup scheme as a KEM; we briefly describe a Cramer-Shoup KEM at the end of this section. The scheme requires a target-collision-resistant hash function $H : G^3 \to \mathbb{Z}/r\mathbb{Z}$ (see Definition 3.1.2) chosen at random from a hash family.

Exercise 23.2.3. Show that the value $v = c^k d^{k\alpha}$ computed in the Encrypt algorithm does satisfy equation (23.1).

Exercise 23.2.4. Show that the tests $u_1, u_2 \in G$ and equation (23.1) imply that $v \in G$.

Exercise 23.2.5. Show that the final stage of Decrypt in the Cramer-Shoup scheme can be efficiently performed using multiexponentiation as

$$\mathsf{m} = e u_1^{r-z_1} u_2^{r-z_2}.$$

[2] Unless performing "bit by bit" encryption, which is a design approach not considered in this book.

KeyGen(κ): Generate a group G of prime order r as in Figure 23.1. Choose random $g_1, g_2 \in G - \{1\}$. Choose integers $0 \leq x_1, x_2, y_1, y_2, z_1, z_2 < r$ uniformly at random and set

$$c = g_1^{x_1} g_2^{x_2}, \quad d = g_1^{y_1} g_2^{y_2}, \quad h = g_1^{z_1} g_2^{z_2}.$$

Choose a target-collision-resistant hash function H. The public key is $\mathsf{pk} = (G, H, g_1, g_2, c, d, h)$. The private key is $\mathsf{sk} = (x_1, x_2, y_1, y_2, z_1, z_2)$.

Encrypt(m, pk): Choose $0 \leq k < r$ uniformly at random, compute $u_1 = g_1^k, u_2 = g_2^k, e = h^k \mathsf{m}, \alpha = H(u_1, u_2, e)$ and $v = c^k d^{k\alpha \pmod r}$. The ciphertext is $\mathsf{c} = (u_1, u_2, e, v)$.

Decrypt($u_1, u_2, e, v, \mathsf{sk}$): First check that $u_1, u_2, e \in G$ and output $\perp$ if this is not the case. Next, compute $\alpha = H(u_1, u_2, e)$ and check whether

$$v = u_1^{x_1 + y_1 \alpha \pmod r} u_2^{x_2 + y_2 \alpha \pmod r}. \tag{23.1}$$

Return $\perp$ if this condition does not hold. Otherwise output

$$\mathsf{m} = e u_1^{-z_1} u_2^{-z_2}.$$

Figure 23.3: Basic Cramer-Shoup Encryption Scheme.

Example 23.2.6. Let $p = 311$, $r = 31$ and denote by G the subgroup of order r in $\mathbb{F}_p^*$. Take $g_1 = 169$ and $g_2 = 121$. Suppose $(x_1, x_2, y_1, y_2, z_1, z_2) = (1, 2, 3, 4, 5, 6)$ so that the public key is

$$(g_1, g_2, c, d, h) = (169, 121, 13, 260, 224).$$

To encrypt $\mathsf{m} = 265 \in G$ choose, say, $k = 15$ and set $u_1 = g_1^k = 24, u_2 = g_2^k = 113$ and $e = \mathsf{m} h^k = 126$. Finally, we must compute α. Since we don't want to get into the details of H, suppose $\alpha = H(u_1, u_2, e) = 20$ and so set $v = c^k d^{k\alpha \pmod r} = c^{15} d^{21} = 89$. The ciphertext is $(u_1, u_2, e, v) = (24, 113, 126, 89)$.

To decrypt one first checks that $u_1^r = u_2^r = e^r = 1$. Then one recomputes α and checks equation (23.1). Since

$$u_1^{x_1 + y_1 \alpha \pmod r} u_2^{x_2 + y_2 \alpha \pmod r} = u_1^{30} u_2^{20} = 89$$

the ciphertext passes this test. One then computes $e u_1^{r - z_1} u_2^{r - z_2} = 126 \cdot 24^{26} \cdot 113^{25} = 265$.

Exercise 23.2.7. Using the same private keys as Example 23.2.6, which of the following ciphertexts are valid, and for those that are, what is the corresponding message? Assume that $H(243, 83, 13) = 2$.

$$(243, 83, 13, 97), \ (243, 83, 13, 89), \ (243, 83, 13, 49).$$

We now turn to the security analysis. Note that the condition of equation (23.1) does not imply that the ciphertext (u_1, u_2, e, v) was actually produced by the Encrypt algorithm. However, we now show that, if u_1 and u_2 are not of the correct form, then the probability that a randomly chosen v satisfies this condition is $1/r$. Indeed, Lemma 23.2.8 shows that an adversary who can solve the discrete logarithm problem cannot even construct an invalid ciphertext that satisfies this equation with probability better than $1/r$.

Lemma 23.2.8. *Let G be a cyclic group of prime order r. Let $g_1, g_2, c, d, v \in G$ and $\alpha \in \mathbb{Z}/r\mathbb{Z}$ be fixed. Suppose $u_1 = g_1^k$ and $u_2 = g_2^{k + k'}$ where $k' \not\equiv 0 \pmod r$. Then the probability, over all choices (x_1, x_2, y_1, y_2) such that $c = g_1^{x_1} g_2^{x_2}$ and $d = g_1^{y_1} g_2^{y_2}$, that $v = u_1^{x_1 + \alpha y_1} u_2^{x_2 + \alpha y_2}$ is $1/r$.*

Proof: Write $g_2 = g_1^w, c = g_1^{w_1}$ and $d = g_1^{w_2}$ for some $0 \leq w, w_1, w_2 < r$ with $w \neq 0$. The values c and d imply that $x_1 + wx_2 = w_1$ and $y_1 + wy_2 = w_2$. Now $u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$ equals g_1 to the power

$$\begin{aligned} k(x_1 + \alpha y_1) + (k + k')w(x_2 + \alpha y_2) &= k((x_1 + wx_2) + \alpha(y_1 + wy_2)) + k'w(x_2 + \alpha y_2) \\ &= k(w_1 + \alpha w_2) + k'w(x_2 + \alpha y_2). \end{aligned}$$

The values $w, w_1, w_2, k, k', \alpha$ are all uniquely determined but, by Lemma 23.2.1, x_2 and y_2 can take any values between 0 and $r-1$. Hence, $u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2}$ equals any fixed value v for exactly r of the r^2 choices for (x_2, y_2). □

Theorem 23.2.9. *The basic Cramer-Shoup encryption scheme is IND-CCA secure if DDH is hard and if the function H is target-collision-resistant.*

Proof: (Sketch) Let A be an adversary against the Cramer-Shoup scheme. Given a DDH instance (g_1, g_2, u_1, u_2) the simulator performs the KeyGen algorithm using the given values g_1, g_2. Hence, the simulator knows the private key $(x_1, x_2, y_1, y_2, z_1, z_2)$. The simulator runs A with this public key.

The algorithm A makes decryption queries and these can be answered correctly by the simulator since it knows the private key. Eventually, A outputs two messages $(\mathsf{m}_0, \mathsf{m}_1)$ and asks for a challenge ciphertext. The simulator chooses a random $b \in \{0, 1\}$, computes $e = u_1^{z_1} u_2^{z_2} \mathsf{m}_b$, $\alpha = H(u_1, u_2, e)$ and $v = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$. Here, and throughout this proof, u_1 and u_2 denote the values in the DDH instance that was given to the simulator. The simulator returns

$$\mathsf{c}^* = (u_1, u_2, e, v).$$

to A. The adversary A continues to make decryption queries, which are answered as above. Eventually, A outputs a bit b'. The simulator returns "valid" as the answer to the DDH instance if $b = b'$ and "invalid" otherwise.

The central idea is that if the input is a valid DDH tuple then c^* is a valid encryption of m_b and so A ought to be able to guess b correctly with non-negligible probability. On the other hand, if the input is not a valid DDH tuple then, by Lemma 23.2.1, $u_1^{z_1} u_2^{z_2}$ could be any element in G (with equal probability) and so c^* could be an encryption of any message $\mathsf{m} \in G$. Hence, given c^*, both messages m_0 and m_1 are equally likely and so the adversary can do no better than output a random bit. (Of course, A may actually output a fixed bit in this case, such as 0, but this is not a problem since b was randomly chosen.)

There are several subtleties remaining in the proof. First, by Lemma 23.2.8, before the challenge ciphertext has been received there is a negligible probability that a ciphertext that was not produced by the Encrypt algorithm satisfies equation (23.1). Hence, the simulation is correct with overwhelming probability. However, the challenge ciphertext is potentially an example of a ciphertext that satisfies equation (23.1) and yet is not a valid output of the algorithm Encrypt. It is necessary to analyse the probability that A can somehow produce another ciphertext that satisfies equation (23.1) without just running the Encrypt algorithm. The target-collision-resistance of the hash function enters at this point (since a ciphertext of the form (u_1, u_2, e', v) such that $H(u_1, u_2, e') = H(u_1, u_2, e)$ would pass the test). Due to lack of space we refer to Section 4 of [159] (for a direct proof) or Section 6.2 of [161] (for a proof using "game hopping"). □

A number of variants of the basic scheme are given by Cramer and Shoup [161] and other authors. In particular, one can design a KEM based on the Cramer-Shoup scheme (see Section 9 of [161]): just remove the component e of the ciphertext and set the encapsulated key to be $K = \mathsf{kdf}(g_1^k, h^k)$. An alternative KEM (with even shorter ciphertext)

was proposed by Kurosawa and Desmedt [359]. Their idea was to set $K = \mathsf{kdf}(v)$ where $v = c^k d^{\alpha k}$ for $\alpha = H(u_1, u_2)$. The KEM ciphertext is therefore just $(u_1, u_2) = (g_1^k, g_2^k)$. The security again follows from Lemma 23.2.8: informally, querying the decryption oracle on badly formed (u_1, u_2) gives no information about the key K.

Exercise 23.2.10. Write down a formal description of the Cramer-Shoup KEM.

Exercise 23.2.11. Show that an adversary against the Cramer-Shoup scheme who knows any pair (z_1, z_2) such that $h = g_1^{z_1} g_2^{z_2}$ can decrypt valid ciphertexts.

Exercise 23.2.12. Suppose an adversary against the Cramer-Shoup scheme knows x_1, x_2, y_1, y_2. Show how the adversary can win the OWE-CCA security game.

Exercise 23.2.13. Suppose the checks that $u_1, u_2 \in G$ are omitted in the Cramer-Shoup cryptosystem. Suppose $G \subset \mathbb{F}_p^*$ where $l \mid (p-1)$ is a small prime (say $l < 2^{10}$). Suppose the Decrypt algorithm uses the method of Exercise 23.2.5. Show how to determine, using a decryption oracle, $z_1 \pmod l$ and $z_2 \pmod l$. Show that if $p-1$ has many such small factors l then one could recover the values z_1 and z_2 in the private key of the Cramer-Shoup scheme.

Cramer and Shoup [160] have shown how the above cryptosystem fits into a general framework for constructing secure encryption schemes using "universal hash proof systems". We do not have space to present this topic.

23.3 Other Encryption Functionalities

There are many variants of public key encryption (such as threshold decryption, server-aided decryption, etc). In this section we briefly sketch two important variants: homomorphic encryption and identity-based encryption.

23.3.1 Homomorphic Encryption

Let $\mathsf{c}_1, \ldots, \mathsf{c}_k$ be ciphertexts that are encryptions under some public key of messages $\mathsf{m}_1, \ldots, \mathsf{m}_k$. The goal of homomorphic encryption is for any user to be able to efficiently compute a ciphertext that encrypts $F(\mathsf{m}_1, \ldots, \mathsf{m}_k)$ for any function F, given only a description of the function F and the ciphertexts $\mathsf{c}_1, \ldots, \mathsf{c}_k$. An encryption scheme that has this property is called **fully homomorphic**.

Homomorphic encryption schemes allow third parties to perform computations on encrypted data. A common additional security requirement is that the resulting ciphertexts do not reveal to a user with the private key what computation was performed (except its result). A typical application of homomorphic encryption is voting: If users encrypt either 0 or 1 under a certain public key[3] then a trusted third party can compute a ciphertext that is an encryption of the sum of all the users' votes, and then this ciphertext can be decrypted to give the total number of votes. If the user with the private key never sees the individual votes then they cannot determine how an individual user voted. A general survey on homomorphic encryption that gives some references for applications is Fontaine and Galand [208].

For many applications it is sufficient to consider encryption schemes that only allow a user to compute $F(\mathsf{m}_1, \ldots, \mathsf{m}_k)$ for certain specific functions (for example, addition in the voting application). In this section we focus on the case where $F(\mathsf{m}_1, \mathsf{m}_2)$ is a group operation.

[3]It is necessary for users to prove that their vote lies in $\{0, 1\}$.

Definition 23.3.1. Let G be a group (written multiplicatively). A public key encryption scheme with message space G and ciphertext space C is said to be **homomorphic** for the group G if there is some efficiently computable binary operation $\star$ on C such that, for all $\mathsf{m}_1, \mathsf{m}_2 \in G$, if c_1 is an encryption of m_1 and c_2 is an encryption of m_2 (both with respect to the same public key) then $\mathsf{c}_1 \star \mathsf{c}_2$ is an encryption of $\mathsf{m}_1\mathsf{m}_2$.

Exercise 23.3.2 shows that one cannot have CCA security when using homomorphic encryption. Hence, the usual security requirement of a homomorphic encryption scheme is that it should have IND-CPA security.

Exercise 23.3.2. Show that a homomorphic encryption scheme does not have IND-CCA security.

Exercise 23.3.3. Let $G = \langle g \rangle$ where g is an element of order r in a group. Let $\mathsf{c}_1 = (\mathsf{c}_{1,1}, \mathsf{c}_{1,2}) = (g^{k_1}, \mathsf{m}_1 h^{k_1})$ and $\mathsf{c}_2 = (\mathsf{c}_{2,1}, \mathsf{c}_{2,2}) = (g^{k_2}, \mathsf{m}_2 h^{k_2})$ be classic textbook Elgamal encryptions of $\mathsf{m}_1, \mathsf{m}_2 \in G$. Define $\mathsf{c}_1 \star \mathsf{c}_2 = (\mathsf{c}_{1,1}\mathsf{c}_{2,1}, \mathsf{c}_{1,2}\mathsf{c}_{2,2})$. Show that $\mathsf{c}_1 \star \mathsf{c}_2$ is an encryption of $\mathsf{m}_1\mathsf{m}_2$ and hence that classic textbook Elgamal encryption is homomorphic for the group G.

Exercise 23.3.4. Let $G = \mathbb{F}_2^l \cong \{0,1\}^l$. Note that G is a group under addition modulo 2 (equivalently, under exclusive-or $\oplus$). For $1 \leq i \leq 2$ let $\mathsf{c}_i = (\mathsf{c}_{i,1}, \mathsf{c}_{i,2}) = (g^{k_i}, \mathsf{m}_i \oplus H(h^{k_i}))$ be semi-textbook Elgamal encryptions of messages $\mathsf{m}_i \in G$. Consider the operation $\mathsf{c}_1 \star \mathsf{c}_2 = (\mathsf{c}_{1,1}\mathsf{c}_{2,1}, \mathsf{c}_{1,2} \oplus \mathsf{c}_{2,2})$. Show that semi-textbook Elgamal is not homomorphic with respect to this operation.

Exercise 23.3.5. A variant of Elgamal encryption that is homomorphic with respect to addition is to encrypt m as $(\mathsf{c}_1 = g^k, \mathsf{c}_2 = g^{\mathsf{m}} h^k)$. Prove that if $(\mathsf{c}_{i,1}, \mathsf{c}_{i,2})$ are ciphertexts encrypting messages m_i for $i = 1, 2$ then $(\mathsf{c}_{1,1}\mathsf{c}_{2,1}, \mathsf{c}_{1,2}\mathsf{c}_{2,2})$ encrypts $\mathsf{m}_1 + \mathsf{m}_2$. Give a decryption algorithm for this system and explain why it is only practical when the messages m are small integers. Hence show that this scheme does not strictly satisfy Definition 23.3.1 when the order of g is large.[4]

23.3.2 Identity-Based Encryption

Section 22.4 briefly mentioned identity-based signatures. Recall that in identity-based cryptography a user's public key is defined to be a function of their "identity" (for example, their email address). There is a master public key. Each user obtains their private key from a key generation center (which possesses the master secret).

In this section we sketch the **basic Boneh-Franklin scheme** [80] (the word "basic" refers to the fact that this scheme only has security against a chosen plaintext attack). The scheme uses pairing groups (see Definition 22.2.14 and Chapter 26). Hence, let G_1, G_2 and G_T be groups of prime order r and let $e : G_1 \times G_2 \to G_T$ be a non-degenerate bilinear pairing.

The first task is to determine the master keys, which are created by the key generation center. Let $g \in G_2$ have order r. The key generation center chooses $1 \leq s < r$ and computes $g' = g^s$. The master public key is (g, g') and the master private key is s. The scheme also requires hash functions $H_1 : \{0,1\}^* \to G_1$ and $H_2 : G_T \to \{0,1\}^l$ (where l depends on the security parameter). The message space will be $\{0,1\}^l$ and the ciphertext space will be $G_2 \times \{0,1\}^l$.

The public key of a user with identity $\mathsf{id} \in \{0,1\}^*$ is $Q_{\mathsf{id}} = H_1(\mathsf{id}) \in G_1$. With overwhelming probability $Q_{\mathsf{id}} \neq 1$, in which case $e(Q_{\mathsf{id}}, g) \neq 1$. The user obtains the

[4] The order of g must be large for the scheme to have IND-CPA security.

private key

$$Q'_{\mathsf{id}} = H_1(\mathsf{id})^s$$

from the key generation center.

To Encrypt a message $\mathsf{m} \in \{0,1\}^l$ to the user with identity id one obtains the master key (g, g'), computes $Q_{\mathsf{id}} = H_1(\mathsf{id})$, chooses a random $1 \leq k < r$ and computes $\mathsf{c}_1 = g^k, \mathsf{c}_2 = \mathsf{m} \oplus H_2(e(Q_{\mathsf{id}}, g')^k)$. The ciphertext is $(\mathsf{c}_1, \mathsf{c}_2)$.

To Decrypt the ciphertext $(\mathsf{c}_1, \mathsf{c}_2)$ the user with private key Q'_{id} computes

$$\mathsf{m} = \mathsf{c}_2 \oplus H_2(e(Q'_{\mathsf{id}}, \mathsf{c}_1)).$$

This completes the description of the basic Boneh-Franklin scheme.

Exercise 23.3.6. Show that the Decrypt algorithm does compute the correct message when $(\mathsf{c}_1, \mathsf{c}_2)$ are the outputs of the Encrypt algorithm.

Exercise 23.3.7. Show that the basic Boneh-Franklin scheme does not have IND-CCA security.

The security model for identity-based encryption takes into account that an adversary can ask for private keys on various identities. Hence, the IND security game allows an adversary to output a challenge identity id^* and two challenge messages $\mathsf{m}_0, \mathsf{m}_1$. The adversary is not permitted to know the private key for identity id^* (though it can receive private keys for any other identities of its choice). The adversary then receives an encryption with respect to identity id^* of m_b for randomly chosen $b \in \{0,1\}$ and must output a guess for b.

Exercise 23.3.8. Suppose there is an efficiently computable group homomorphism $\psi : G_2 \to G_1$. Show that if an adversary knows ψ and can compute preimages of the hash function H_1 then it can determine the private key for any identity by making a private key query on a different identity.

If the output of H_2 is indistinguishable from random l-bit strings then it is natural to believe that obtaining the message from a ciphertext under a passive attack requires computing

$$e(Q'_{\mathsf{id}}, \mathsf{c}_1) = e(Q^s_{\mathsf{id}}, g^k) = e(Q_{\mathsf{id}}, g)^{sk}.$$

Hence, it is natural that the security (at least, in the random oracle model) depends on the following computational problem.

Definition 23.3.9. Let G_1, G_2 and G_T be groups of prime order r and let $e : G_1 \times G_2 \to G_T$ be a non-degenerate bilinear pairing. The **bilinear Diffie-Hellman problem** (**BDH**) is: Given $Q \in G_1$, $g \in G_2$, g^a and g^b, where $1 \leq a, b < r$, to compute

$$e(Q, g)^{ab}.$$

Exercise 23.3.10. Show that if one can solve CDH in G_2 or in G_T then one can solve BDH.

As seen in Exercise 23.3.7, the basic Boneh-Franklin scheme does not have IND-CCA security. To fix this one needs to provide some extra components in the ciphertext. Alternatively, one can consider the basic Boneh-Franklin scheme as an identity-based KEM: The ciphertext is $\mathsf{c}_1 = g^k$ and the encapsulated key is $K = \mathsf{kdf}(e(Q_{\mathsf{id}}, g')^k)$. In the random oracle model (treating both H_1 and kdf as random oracles) one can show that the Boneh-Franklin identity-based KEM has IND-CCA security (in the security model for

identity-based encryption as briefly mentioned above) assuming that the BDH problem is hard. We refer to Boneh and Franklin [80, 81] for full details and security proofs.

There is a large literature on identity-based encryption and its extensions, including schemes that are secure in the standard model. We do not discuss these topics further in this book.