**Lecture Notes for CS-578 Cryptography (SS2006)** Prof. Michael Backes

# 7. Public-key Encryption, Diffie-Hellman, ElGamal

Lecture 10 Saarland University

This is our first chapter on *public-key encryption*, also called *asymmetric encryption*. Each party has a pair of keys: The *public key* is known to everybody and is used for encryption, while the *secret/private key* is kept secret and is used for decryption. Thus everybody can encrypt messages with the public key while only the owner of the corresponding secret key can decrypt them. Since only public keys have to be transmitted, key distribution is considerably simplified as only the authenticity of exchanged keys is relevant. This is an important difference with respect to distribution of symmetric keys where also the privacy of exchanged keys has to be guaranteed.

## 7.1 Definition of Public-key Encryption Schemes

The next definition formalizes the notion of public-key encryption schemes:

**Definition 7.1 (Public-key Encryption Scheme)** *A public-key encryption scheme consists of three efficient algorithms* $(\mathsf{Gen}, \mathsf{E}, \mathsf{D})$*, which are defined as follows:*

- *The randomized* key generation *algorithm* $\mathsf{Gen}$ *takes the security parameter in unary representation (for defining complexity in the length of the security parameter) as input and returns a pair* $(pk, sk)$ *of keys, the* public key $pk$ *and the matching* secret key $sk$.
- *The* encryption algorithm $\mathsf{E}$ *takes the public key* $pk$ *and a plaintext* $m$ *and returns a ciphertext* $c$. *This algorithm may be randomized.*
- *The deterministic* decryption algorithm $\mathsf{D}$ *takes the secret key* $sk$ *and a ciphertext* $c$ *and returns a plaintext* $m$.
- *The* message space $\mathcal{M}_{pk}$ *associated to a public key* $pk$ *is the set of plaintexts* $m$ *for which* $\mathsf{E}(pk, m)$ *never returns a distinguished error symbol* $\downarrow$. *We require that, for any key-pair* $(pk, sk)$ *that might be output by* $\mathsf{Gen}$ *and for any message* $m \in \mathcal{M}_{pk}$*, if* $c \leftarrow \mathsf{E}(pk, m)$ *then* $m = \mathsf{D}(sk, c)$.

◇

We assume that all algorithms (including challengers and adversaries in the following) get the security parameter in unary representation as input so that the efficiency of such algorithms can be meaningfully measured in the security parameter. We don't write this explicitly in the following to increase readability.

## 7.2 Semantic Security of Public-key Encryption Schemes against CPA

Security of public-key encryption schemes against chosen-plaintext attack (passive adversaries) is defined similar to the symmetric case.

**Definition 7.2 (CPA Challenger)** *Let* $\mathsf{PubEnc} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ *be a public-key encryption scheme, and let* $n$ *be the security parameter. The* CPA challenger *for* PubEnc *and* $n$ *is defined as follows:*

- *First, it chooses a bit* $b$*, creates keys* $(pk, sk) \leftarrow \mathsf{Gen}(n)$*, and outputs* $pk$*.*
- *It then receives two plaintexts* $m_0, m_1 \in \mathcal{M}_{pk}$ *with* $|m_0| = |m_1|$*, computes an encryption* $c \leftarrow \mathsf{E}(pk, m_b)$*, and outputs* $c$*.*

◇

An adversary wins the game against the CPA challenger if, intuitively, it is able to deduce the bit $b$ better than by pure guessing. In the following, $Exp_{\mathsf{A}}^{\mathsf{CPA}}(b)$ denotes the experiment where the adversary A interacts with the CPA challenger whose bit is chosen as $b$. Furthermore $Exp_{\mathsf{A}}^{\mathsf{CPA}}(b) = 0$ and $Exp_{\mathsf{A}}^{\mathsf{CPA}}(b) = 1$ denote the event that the adversary outputs 0 and 1 in the respective experiment.

The advantage can now be defined as usual. Note that the security parameter $n$ induces a uniform sequence of public-key encryption schemes, each of which is defined for a fixed value of $n$, as well as a uniform sequence of CPA challengers and adversaries. Thus, the public-key encryption algorithms Gen, E, D, the challenger CPA, and the adversary A constitute individual algorithms that are parameterized by the security parameter. Hence the advantage naturally also constitutes a function of $n$. (This might be clearer for public-key encryption than it was for symmetric encryption, since key generation as well as encryption and decrypting algorithms are defined for arbitrary values of $n$ already. This stands in contrast to, e.g., AES, which was only defined for fixed key sizes, and where the definition of sequences consequently might have looked artificial.)

**Definition 7.3 (CPA Advantage)** *Let* $\mathsf{PubEnc} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ *be a public-key encryption scheme. The* advantage *of an adversary* A *against the CPA challenger for* E *is defined as follows (as a function of the security parameter* $n$ *since* PubEnc*, the* CPA *challenger, and* A *are parameterized in* $n$*):*

$$Adv^{\mathsf{CPA}}[\mathsf{A}, \mathsf{PubEnc}](n) := \left| \Pr\left[ Exp_{\mathsf{A}}^{\mathsf{CPA}}(0) = 1 \right] - \Pr\left[ Exp_{\mathsf{A}}^{\mathsf{CPA}}(1) = 1 \right] \right|.$$

◇

**Definition 7.4 (Semantic Security of Public-key Encryption against CPA)** *A public-key encryption scheme* $\mathsf{PubEnc} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ *is* semantically secure against chosen-plaintext attack (CPA) *if for all efficient adversaries* A*, the advantage* $Adv^{\mathsf{CPA}}[\mathsf{A}, \mathsf{PubEnc}](n)$ *is negligible in* $n$*.* ◇

## 7.3 Semantic Security of Public-key Encryption Schemes against Active Adversaries (CCA2)

Security against active adversaries is defined similarly, but the adversary is given the possibility to additionally decrypt any ciphertexts it wants. Note that there are two variants of this. For *non-adaptive chosen-ciphertext attack* or *lunchtime attack*, the adversary is allowed to decrypt messages only before selecting the messages $m_0, m_1$. For *adaptive chosen-ciphertext attack* the adversary may ask for decryptions even after receiving the challenge-ciphertext, except that he is forbidden to ask for the decryption of the challenge-ciphertext itself. The latter notion is more powerful and constitutes the standard security definition of public-key encryption in modern cryptography. It is abbreviated CCA2.

**Definition 7.5 (CCA2 Challenger)** *Let* $\mathsf{PubEnc} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ *be a public-key encryption scheme,* $n$ *be the security parameter, and* $\downarrow$ *denote an arbitrary error symbol that is not in the range of* E *for* *any $pk$ and $m$. The* $\mathsf{CCA2}$ challenger *for* $\mathsf{PubEnc}$ *and $n$ maintains a variable $c^*$ and is defined as follows:*

- *First, it chooses a bit $b$, sets $c^* := \downarrow$, creates keys $(pk, sk) \leftarrow \mathsf{Gen}(n)$, and outputs $pk$.*
- *If it receives a decryption query $c$ it does the following: If $c \neq c^*$, it computes $m := \mathsf{D}(sk, c)$ and outputs $m$.*
- *If it receives two plaintexts $m_0, m_1 \in \mathcal{M}_{pk}$ with $|m_0| = |m_1|$, it computes an encryption $c \leftarrow \mathsf{E}(pk, m_b)$, sets $c^* := c$, and outputs $c$.*

$\diamond$

An adversary wins the game against the $\mathsf{CCA2}$ challenger if it is able to deduce the bit $b$ better than by pure guessing. In the following, $Exp_{\mathsf{A}}^{\mathsf{CCA2}}(b)$ denotes the experiment where the adversary $\mathsf{A}$ interacts with the $\mathsf{CCA2}$ challenger whose bit is chosen as $b$. Furthermore $Exp_{\mathsf{A}}^{\mathsf{CCA2}}(b) = 0$ and $Exp_{\mathsf{A}}^{\mathsf{CCA2}}(b) = 1$ denote the event that the adversary outputs 0 and 1 in the respective experiment.

The advantage is then defined as a function of $n$ as usual.

**Definition 7.6 (CCA2 Advantage)** *Let $\mathsf{PubEnc} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ be a public-key encryption scheme. The* advantage *of an adversary $\mathsf{A}$ against the CCA2 challenger for $\mathsf{PubEnc}$ is defined as follows (as a function of the security parameter $n$ again since $\mathsf{PubEnc}$, the $\mathsf{CCA2}$ challenger, and $\mathsf{A}$ are parameterized in $n$):*

$$Adv^{\mathsf{CCA2}}[\mathsf{A}, \mathsf{PubEnc}](n) := \left| \mathsf{Pr}\left[ Exp_{\mathsf{A}}^{\mathsf{CCA2}}(0) = 1 \right] - \mathsf{Pr}\left[ Exp_{\mathsf{A}}^{\mathsf{CCA2}}(1) = 1 \right] \right|.$$

$\diamond$

**Definition 7.7 (Semantic Security of Public-key Encryption against CCA2)** *A public-key encryption scheme $\mathsf{PubEnc} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ is* semantically secure against adaptive chosen-ciphertext attack ($\mathsf{CCA2}$) *if for all efficient adversaries $\mathsf{A}$, the advantage $Adv^{\mathsf{CCA2}}[\mathsf{A}, \mathsf{PubEnc}](n)$ is negligible in $n$.*

$\diamond$

## 7.4 The Discrete Logarithm and Diffie-Hellman Problems

Next we will define three problems that are conjectured to be hard in certain groups. They can be defined in arbitrary (cyclic) groups. In practice, however, they are most often used in $\mathbb{Z}_p^*$ for a prime $p$ or in subgroups of prime order $q$ of $\mathbb{Z}_p^*$ where $p$ is a prime as well and $q|p-1$. We will denote such subgroups of $\mathbb{Z}_p^*$ by $G_q$. The following definitions will be specifically given for $\mathbb{Z}_p^*$ and for subgroups $G_q$ of $\mathbb{Z}_p^*$, i.e., consider a group $G$ which is either $G_q$ or $\mathbb{Z}_p^*$ for some $p, q$. For $g \in G$ one defines the *discrete exponentiation* as

$$\mathsf{DExp}_g : \mathbb{N} \to G, \quad x \mapsto g^x.$$

The *discrete logarithm* is defined as

$$\mathsf{DLog}_g : G \to \mathbb{N}, \quad h \mapsto x.$$

where $x \in \mathbb{N}$ is the smallest natural number such that $g^x = h$, with $\mathsf{DLog}_g(h) := \infty$ if no such $x$ exists. The discrete logarithm $\mathsf{DLog}$ is conjectured to be hard to compute in $G$, and in many more groups as well. For a generator $g$ of $G$, the *Diffie-Hellman function* is defined as

$$\mathsf{DH}_g : G \times G \to G, \quad (g^x, g^y) \mapsto g^{xy}.$$

The *computational Diffie-Hellman (CDH) problem* is defined as follows: Let $x, y \leftarrow_{\mathcal{R}} \{1, \dots, |G|\}$ and $g$ a generator of $G$, compute the Diffie-Hellman function $\mathsf{DH}(g^x, g^y)$, i.e., given $g^x, g^y$, compute $g^{xy}$. This is conjectured to be hard in $G$. This in particular led to the Diffie-Hellman key exchange protocol and to the ElGamal encryption scheme, which we will describe later.

It is easy to see that if the discrete logarithm is easy to compute, then the computational Diffie-Hellman problem is easy to solve. In other words, solving the CDH problem is at most as difficult as computing the discrete logarithm. The converse direction is, in general, unknown.

The *Decisional Diffie-Hellman (DDH) problem* is closely related to the CDH problem. The difference is that the adversary is not required to compute the value $g^{xy}$, but he needs to distinguish it from a value $g^z$ for a random $z$. This assumption will play an important role in the sequel, thus we define it formally, again in terms of a game. We however only define it for subgroups $G_q$ of $\mathbb{Z}_p^*$, where the DDH problem is indeed considered hard; in $\mathbb{Z}_p^*$ the DDH problem can be efficiently solved, see below.

For a rigorous definition of the game, the following technical subtlety arises: The group $G_q$ is a subgroup of $\mathbb{Z}_p^*$, so strictly speaking we needed two "security parameters" $n$ and $n^*$ measuring the size of $p$ and $q$, respectively. We avoid this by saying that the bit-length of $q$ is the primary security parameter $n$, and that $n^*$ can be derived from $n$ via a public polynomially bounded function $n_p(\cdot)$, i.e., $n^* := n_p(n)$.

**Definition 7.8 (DDH Challenger)** *The* DDH *challenger (for subgroups $G_q$ of $\mathbb{Z}_p^*$) for a given security parameter $n$ is defined as follows:*

- *Firstly, it randomly chooses an $n$-bit prime $q$ and an $n_p(n)$-bit prime $p$ such that $q|p-1$. After that, it randomly chooses an element $g \in \mathbb{Z}_p^*$ of order $q$.*
- *Secondly, it randomly chooses a bit $b$ and values $x, y, z \leftarrow_{\mathcal{R}} \{1, \dots, q\}$. Depending on the value of $b$, it outputs*
  - $(q, p, g, g^x, g^y, g^{xy})$ *if* $b = 0$.
  - $(q, p, g, g^x, g^y, g^z)$ *if* $b = 1$.

$\diamond$

An adversary wins the game against the DDH challenger if it is able to deduce the bit $b$ significantly better than by pure guessing. Again, the adversary advantage captures how much better an adversary can do than to purely guess the bit. In the following, $Exp_{\mathsf{A}}^{\mathsf{DDH}}(b)$ denotes the experiment where the adversary $\mathsf{A}$ interacts with the DDH challenger whose bit is chosen as $b$. Furthermore $Exp_{\mathsf{A}}^{\mathsf{DDH}}(b) = 0$ and $Exp_{\mathsf{A}}^{\mathsf{DDH}}(b) = 1$ denote the event that the adversary outputs 0 and 1 in the respective experiment. The advantage is defined as usual.

**Definition 7.9 (DDH Advantage)** *The advantage of an adversary* $\mathsf{A}$ *against the DDH challenger is defined as follows (as a function of the security parameter $n$ again):*

$$Adv^{\mathsf{DDH}}[\mathsf{A}](n) := \left| \mathsf{Pr}\left[ Exp_{\mathsf{A}}^{\mathsf{DDH}}(0) = 1 \right] - \mathsf{Pr}\left[ Exp_{\mathsf{A}}^{\mathsf{DDH}}(1) = 1 \right] \right|.$$

$\diamond$

**Assumption 1 (Hardness of the DDH Problem in $G_q$)** *The DDH problem in groups $G_q$ as constructed above is conjectured to be hard, i.e., it is conjectured that $Adv^{\mathsf{DDH}}[\mathsf{A}](n)$ is negligible in the security parameter $n$.*

It is again easy to see that, if the CDH problem is easy, i.e., if one can compute $g^{xy}$ given $g^x$ and $g^y$, then the DDH problem can be decided efficiently by comparing the computed value with the given value. The converse is not known in general, but we will see now that in $\mathbb{Z}_p^*$ the DDH problem is easy to solve, while the CDH problem is conjectured to be hard. To see that the DDH problem is easy to solve in $\mathbb{Z}_p^*$, we make the following observation: We have seen in Remark 1 in Section 6.1.3 that if $g$ is a generator of $\mathbb{Z}_p^*$, then $g^r$ is a quadratic residue (QR) iff $r$ is even; hence for $r$ randomly chosen from $\{1,\ldots,p-1\}$, $g^r$ is a QR with probability $\frac{1}{2}$. However, if $x, y \leftarrow_{\mathcal{R}} \{1,\ldots,p-1\}$ then the probability that $g^{xy}$ is a QR is precisely $\frac{3}{4}$.

Thus an adversary can do the following: receiving $(q, p, g, h, i, k)$ from the DDH challenger, the adversary outputs 0 if $k$ is a QR, and 1 if $k$ is not a QR. (Note that this can be easily computed using the Legendre symbol.) Then one easily calculates

$$\mathsf{Pr}\left[Exp_{\mathsf{A}}^{\mathsf{DDH},\mathbb{Z}_p^*}(0) = 0\right] = \frac{3}{4} \quad \text{and} \quad \mathsf{Pr}\left[Exp_{\mathsf{A}}^{\mathsf{DDH},\mathbb{Z}_p^*}(1) = 0\right] = \frac{1}{2},$$

and thus

$$Adv^{\mathsf{DDH},\mathbb{Z}_p^*}[\mathsf{A}] = \left|\frac{3}{4} - \frac{1}{2}\right| = \frac{1}{4}.$$

Before continuing with applications of the above concepts, let us briefly take a look at a property of the $\mathsf{DLog}$ called *random self-reducibility.* It turns out that, if one can compute the $\mathsf{DLog}$ for a small fraction of inputs, then one can compute it for all inputs.

**Lemma 7.1 (Random Self-reducibility of** $\mathsf{DLog}$**)** *Let $g$ be a generator of $\mathbb{Z}_p^*$. Suppose there exists an algorithm $\mathsf{A}$ for computing $\mathsf{DLog}_g(h)$ in time $T$ whenever $h \in S \subseteq \mathbb{Z}_p^*$, where $|S| = \epsilon \cdot |\mathbb{Z}_p^*|$. Then there exists an algorithm $\mathsf{B}$ for computing $\mathsf{DLog}_g(h)$ for all $h$ in expected time $T/\epsilon$.* □

*Proof.* Since $g$ is a generator, the discrete log of $h$ to the base $g$ is an integer $x$ where $0 < x \leq p-1$ and $h = g^x$. The algorithm $\mathsf{B}$ picks a random $y \leftarrow_{\mathcal{R}} \{1,\ldots,p-1\}$, uses $\mathsf{A}$ to compute the discrete logarithm $z$ of $h \cdot g^y$, and tests if the result was correct. It repeats the choice of $y$ and the subsequent testing until a correct discrete logarithm is output. The element $g^y$ is distributed uniformly in $\mathbb{Z}_p^*$ as $y$ is uniformly distributed in $\{1,\ldots,p-1\}$, thus the probability that $h \cdot g^y \in S$ is $\epsilon$. In this case $\mathsf{A}$ computes the logarithm $z$ of $h \cdot g^y$ correctly. $\mathsf{B}$ then simply outputs $z - y \bmod p-1$ as the discrete logarithm of $h$. This is correct since

$$g^{z-y} = g^z \cdot g^{-y} = h \cdot g^y \cdot g^{-y} = h.$$

The expected running time of algorithm $\mathsf{B}$ is $T/\epsilon$: it needs to do an expected number of $\frac{1}{\epsilon}$ loops, each of which takes time $T$ (plus a small polynomial overhead in each loop for choosing the values $y$ and testing if the logarithm was correctly computed, which we neglected). ■

## 7.5 Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange was invented by Wittfield Diffie and Martin Hellman in 1976. Using this protocol it is possible for two parties Alice and Bob to agree on a secret value in the presence of a passive adversary observing their entire communication. The algorithm is depicted in Figure 7.1. It was traditionally proposed for $\mathbb{Z}_p^*$ for a large prime $p$ but works in principle in any cyclic group (and is today usually used in subgroups $G_q$ of $\mathbb{Z}_p^*$).

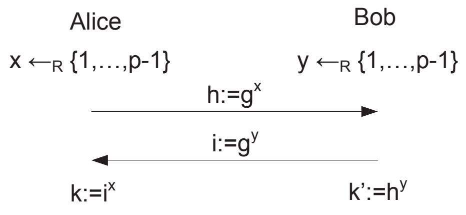

Figure 7.1: Diffie-Hellman Key Exchange

Let $g$ be a generator of $\mathbb{Z}_p^*$ which is publicly known. Alice chooses $x \leftarrow_{\mathcal{R}} \{1, \dots, p-1\}$ and sends $h := g^x \in \mathbb{Z}_p^*$ to Bob, whereas Bob chooses $y \leftarrow_{\mathcal{R}} \{1, \dots, p-1\}$ and sends $i := g^y \in \mathbb{Z}_p^*$ to Alice. Then Alice computes $k := i^x$ and Bob computes $k' := h^y$. Then an easy calculation shows that

$$k = i^x = (g^y)^x = g^{xy} = (g^x)^y = h^y = k'.$$

So both Alice and Bob know a common secret $k \in \mathbb{Z}_p^*$ that they can use do derive keys for other encryption schemes such as AES.

## 7.6 The ElGamal Encryption System

The ElGamal encryption scheme is the first public-key encryption scheme that we will explore. Its security is based on the hardness of the Decisional Diffie-Hellman problem. The ElGamal encryption scheme $\mathsf{E}^{\mathsf{ElGamal}} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ is defined as follows for security parameter $n$:

- Key generation $\mathsf{Gen}$ chooses an $n$-bit prime $q$ and an $n_p(n)$-bit prime $p$ such that $q|p-1$. After that, it randomly chooses an element $g \in \mathbb{Z}_p^*$ of order $q$. Furthermore, it chooses a random value $x \leftarrow_{\mathcal{R}} \{1, \dots, q\}$ and computes $h := g^x$. It sets $pk = (q, p, g, h)$ and $sk = (q, p, g, x)$ and outputs the key-pair $(pk, sk)$.
- Encryption $\mathsf{E}(pk, m)$ for messages $m \in G_q := \langle g \rangle$ is computed as follows: Choose $y \leftarrow_{\mathcal{R}} \{1, \dots, q\}$, compute $i := g^y$, $c' := m \cdot h^y$ and output $c := (i, c')$.
- Decryption $\mathsf{D}(sk, (i, c'))$ works by computing $m := c' \cdot (i^x)^{-1}$.

The correctness of this scheme can be verified as follows: Let $pk = (q, p, g, h = g^x)$, $sk = (q, p, g, x)$ and $m \in G_q$. Then

$$\begin{aligned} \mathsf{D}(sk, \mathsf{E}(pk, m)) &= \mathsf{D}(sk, (g^y, m \cdot h^y)) \\ &= m \cdot h^y \cdot (g^y)^{-x} \\ &= m \cdot g^{xy} \cdot g^{-xy} \\ &= m. \end{aligned}$$

One can easily see that a total break of this scheme, i.e., the ability to compute the secret value $x$ given an encryption $c$, requires solving the discrete logarithm problem, and recovering the plaintext from an encryption requires solving the Computational Diffie-Hellman problem. However, we will see that for semantic security against CPA, it's the hardness of the Decisional Diffie-Hellman problem that we have to be assuming.

**Theorem 7.1 (CPA-Security of ElGamal)** *If the DDH problem for subgroups $G_q$ of $\mathbb{Z}_p^*$ as constructed above is hard, then $\mathsf{E}^{\mathsf{ElGamal}} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ is semantically secure against chosen-plaintext*

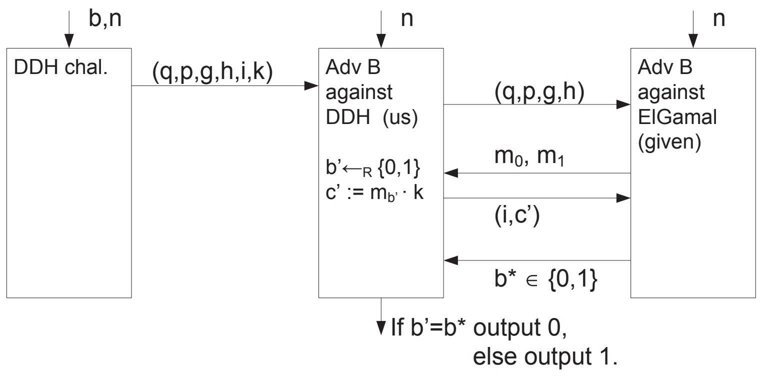

Figure 7.2: The Security of ElGamal against Passive Adversaries

*attack (CPA).* □

*Proof.* Given an adversary $\mathsf{A}$ against the $\mathsf{CPA}$ challenger for ElGamal we construct an adversary $\mathsf{B}$ against the $\mathsf{DDH}$ challenger as depicted in Figure 7.2. We need to calculate

$$Adv^{\mathsf{DDH}}[\mathsf{B}] = \left|\Pr\left[Exp_{\mathsf{B}}^{\mathsf{DDH}}(0) = 1\right] - \Pr\left[Exp_{\mathsf{B}}^{\mathsf{DDH}}(1) = 1\right]\right|. \tag{7.1}$$

First, observe that for the second expression we have

$$\Pr\left[Exp_{\mathsf{B}}^{\mathsf{DDH}}(1) = 1\right] = \frac{1}{2}, \tag{7.2}$$

as the value $k$ output by the DDH challenger in this experiment is $g^z$ for $z$ randomly chosen from $\{1, \ldots, q\}$. Thus $g^z$ is random in $G_q$ and consequently the encryption $c' = m_{b'} \cdot k$ computed by $\mathsf{B}$ constitutes a One-time Pad in $G_q$. Thus in the experiment $Exp_{\mathsf{B}}^{\mathsf{DDH}}(1)$ the adversary $\mathsf{A}$ gains no information on $\mathsf{B}$'s random bit $b'$. Now let us calculate the first expression of the absolute value in Equation 7.1:

$$\begin{aligned}
& \Pr\left[Exp_{\mathsf{B}}^{\mathsf{DDH}}(0) = 1\right] \\
= \ & \Pr\left[b' \neq b^* \mid b = 0\right] && (7.3) \\
= \ & \Pr\left[b' = 1 \wedge b^* = 0 \mid b = 0\right] + \Pr\left[b' = 0 \wedge b^* = 1 \mid b = 0\right] \\
= \ & \frac{1}{2} \cdot \Pr\left[Exp_{\mathsf{A}}^{\mathsf{CPA}}(1) = 0\right] + \frac{1}{2} \cdot \Pr\left[Exp_{\mathsf{A}}^{\mathsf{CPA}}(0) = 1\right] \\
= \ & \frac{1}{2} \cdot \left(1 - \Pr\left[Exp_{\mathsf{A}}^{\mathsf{CPA}}(1) = 1\right]\right) + \frac{1}{2} \cdot \Pr\left[Exp_{\mathsf{A}}^{\mathsf{CPA}}(0) = 1\right] \\
= \ & \frac{1}{2} + \frac{1}{2} \cdot \left(\Pr\left[Exp_{\mathsf{A}}^{\mathsf{CPA}}(0) = 1\right] - \Pr\left[Exp_{\mathsf{A}}^{\mathsf{CPA}}(1) = 1\right]\right) && (7.4)
\end{aligned}$$

Thus we obtain

$$Adv^{\mathsf{DDH}}[\mathsf{B}] = \frac{1}{2} Adv^{\mathsf{CPA}}[\mathsf{A}]$$

which concludes the proof. ■

Thus, the ElGamal encryption scheme is CPA-secure in groups where the DDH problem is conjectured to be hard. Remember that this is however not the case in $\mathbb{Z}_p^*$, and we leave it as a simple exercise to show that CPA-security of ElGamal does not hold in $\mathbb{Z}_p^*$ (this can be shown similar to how we solved the DDH problem in $\mathbb{Z}_p^*$).

However, it turns out that in any group, ElGamal encryption as described above is completely insecure against chosen-ciphertext attack (CCA2), i.e., against active attackers.

**Proposition 7.1 (ElGamal is not CCA2-secure)** *Given the ElGamal encryption scheme* $\mathsf{E}^{\mathsf{ElGamal}} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ *(over an arbitrary group), there exists an adversary* $\mathsf{A}$ *such that*

$$Adv^{\mathsf{CCA2}}[\mathsf{A}, \mathsf{E}^{\mathsf{ElGamal}}] = 1.$$

*Proof.* The adversary $\mathsf{A}$ picks two arbitrary challenge messages $m_0 \neq m_1 \in G$ and asks for an encryption of one of them, yielding a ciphertext $(i, c)$ of $m_b$. The adversary chooses an arbitrary $c'$ and lets the $\mathsf{CCA2}$ challenger decrypt $(i, c')$ yielding a message $m' = c'/i^x$. The adversary then simply computes $i^x = c'/m'$ and $m^* = c/i^x$. By construction, we have $m^* = c/i^x = m_b$ so that the adversary outputs 0 if $m^* = m_0$ and 1 if $m^* = m_1$. ∎

Note that this attack could be countered by testing if a value $i$ was reused. While this might not be practical in applications, there are additionally more sophisticated attacks that circumvent this additional test: Instead of asking the $\mathsf{CCA2}$ challenger to decrypt the ciphertext $(i, c')$, i.e., for the same $i$ that also occurred in the challenge-ciphertext, the adversary can *blind* this value $i$ as follows. The adversary chooses an arbitrary $z \in \{1, \ldots, |G|\}$ as well as an arbitrary $c' \in G$ and lets the $\mathsf{CCA2}$ challenger decrypt $(i^z, c')$. The element $i^z$ is distributed uniformly in $G$, thus the $\mathsf{CCA2}$ challenger cannot tell it apart from a correctly generated $g^r$ for some other $r$. Thus the $\mathsf{CCA2}$ challenger decrypts the ciphertext and returns $m' := c'/i^{xz}$. Then the adversary can compute $k' := i^{xz} = c'/m'$ and $i^x := k'^{(z^{-1} \in G)}$. Finally, he can decrypt $c'$ by computing $m := c'/i^x$ and then proceed as in the previous proof.

Thus, in order to make ElGamal-style encryption schemes secure against chosen-ciphertext attack, one needs a more sophisticated construction, which will be described in the next chapter.