Brief Introduction to Provable Security

Michel Abdalla

Département d'Informatique, École normale supérieure
michel.abdalla@ens.fr
http://www.di.ens.fr/users/mabdalla

1 Introduction

The primary goal of cryptography is to enable parties to communicate securely over an insecure channel, which may be under the control of an adversary. Though originally used mainly for the purpose of protecting the privacy of messages, cryptography now encompasses many other goals, such as guaranteeing the integrity of messages being exchanged or the authenticity of the sender.

For most of its history, cryptography was essentially a game played between designers and attackers in which one side would try to outsmart the other by conceiving ad hoc attack and defense mechanisms for their particular goals [Bel11]. Although this ad hoc aspect may always be present in the field, cryptography has since become a well established science, with clear security definitions and objectives.

The exact security objectives in which one might be interested will be determined by the application one has in mind and may depend on many factors, such as how powerful adversaries may be or the type of information that needs to be protected. The two most common goals usually considered are data privacy and authenticity. While the goal of *data privacy* is to keep unintended parties from learning the contents of the message being sent over the channel, *data authenticity* aims at guaranteeing that the contents of the message have not been tampered with during the course of transmission.

Encryption schemes. The usual way to achieve data privacy and allow parties to exchange messages privately over an insecure channel is to use an *encryption* scheme. In these schemes, there are two types of keys, known as the encryption and decryption keys. Whenever a sender wants to send a private message, or plaintext, to a receiver, he first converts it into an enciphered form, called a ciphertext, with the help of the encryption key and then sends the latter to the receiver. Upon the receipt of a ciphertext, the receiver can use the decryption key to recover the original plaintext. The algorithms used to generate the ciphertext and to recover the plaintext are known as the encryption and decryption algorithms respectively.

The classical example of an encryption scheme is Shannon's one-time pad [Sha49]. In the one-time pad scheme, both the sender and receiver have to share a secret key whose length is at least that of the message being exchanged. To encrypt a message, the sender simply computes the bit-wise XOR of the message with the shared secret key, which serves as the encryption key, and sends it to the receiver. Upon receiving a ciphertext, the receiver can recover the original message in a similar manner using the shared secret key as the decryption key. Despite its simplicity, Shannon showed that the one-time pad is actually as secure as it gets since, as long as the shared secret key does not get reused, it is impossible for an adversary to gain any information about the plaintext, other than its length, from the ciphertext. Moreover, this is the case even if the adversary has unlimited computational resources. Its main drawback lies in the key size, which needs to be at least as long as the message, thus imposing a limit on the amount of information that can be securely transmitted.

Signature schemes. The standard way of achieving data authenticity is to use a *signature* scheme. As in an encryption scheme, there are two types of keys, to which we refer as signing and verification keys. Using the signing key, a sender can create a signature for each message whose authenticity needs to be guaranteed. This signature will be attached to the message before its transmission and will serve as a testament to the fact that the message is authentic. After receiving a message together with its signature, the receiver can verify its authenticity by checking the validity of the signature with the help of the verification key. The algorithms used to generate the signature and to verify its validity are known as the signing and verification algorithms respectively.

Symmetric and asymmetric settings. In order for an encryption scheme to guarantee data privacy, it is clear that the decryption key used by the receiver should be kept hidden from the adversary or else the adversary could use it to recover the plaintext associated with any ciphertext sent over the channel. Likewise, to guarantee data authenticity, the signing key of a signature scheme needs to remain secret or else the adversary could fabricate authenticators for any message of its choice.

As for the encryption and verification keys used in these schemes, they might be either secret or not depending on the particular setting in which we are interested. In the *private-key* cryptographic setting, both encryption and verification keys are assumed to be unknown to the adversary. Due to the secrecy of these keys, both senders and receivers have to agree on their values before they can start communicating securely. Moreover, since these keys usually have high entropy, one may need a cryptographic device to safely store them. In this setting, signature schemes are also known as message authentication codes. On the other hand, in the *public-key* cryptographic setting, encryption and verification keys are not necessarily hidden from the adversary. They are only mathematically related to the corresponding decryption and signing keys and may be made public. Because the encryption and verification keys are usually equal to the decryption and signing keys, respectively, in the private-key setting and different from them in the public-key setting, we also refer to these settings as *symmetric* and *asymmetric*.

The notion of public-key cryptography was introduced by Diffie and Hellman [DH76] and is one of the main innovations of modern cryptography. Unlike private-key cryptography, public-key cryptography does not require previously established secrets to enable secure communication between parties. In public-key cryptography, each user has a pair of *public* and *secret* keys associated to him. While the public key is known to all parties, including the adversary, the secret key should only be known to the user with whom it is associated.

Since public and secret keys are mathematically related, it is always possible for an adversary to compute a corresponding secret key when given a public key. Hence, public-key cryptography should only be considered in an environment in which adversaries are assumed to be computationally bounded. In this *computational-complexity* setting, rather than *impossibility*, we talk about the *infeasibility* of breaking the security of a scheme.

In order to link users to their public keys, public-key encryption and signature schemes have to rely on the existence of a public-key infrastructure (PKI), where a trusted authority certifies the relation between users and their public keys by means of a digital signature. Since users need to know the public key associated with the trusted authority itself to be able to verify the validity of signatures issued by this authority, the latter public key needs to be pre-distributed to all users in the system.

Key exchange. Although public-key encryption and signature schemes can guarantee the privacy and authenticity of exchanged messages when users have access to certified copies of each other's public keys, their costs may be too high for certain applications. To avoid the use of such primitives and improve the overall efficiency, users often prefer to secure their communication via symmetric encryption and signature schemes. Unfortunately, this is only possible when users are in possession of pre-established common secrets. To alleviate the need for pre-established secrets without significantly impacting the overall efficiency, another way of dealing with this problem is for users to first establish a common secret key via a key exchange protocol and to use this key to derive keys for symmetric encryption and signature schemes.

Public information: $(\mathbb{G}, g, p)$

Alice		**Bob**
$x \xleftarrow{R} \mathbb{Z}_p$; $X \leftarrow g^x$		$y \xleftarrow{R} \mathbb{Z}_p$; $Y \leftarrow g^y$
	$\xrightarrow{X}$	
	$\xleftarrow{Y}$	
$SK_A \leftarrow Y^x$		$SK_B \leftarrow X^y$

Figure 1: The Diffie-Hellman key exchange protocol [DH76]. The protocol works over a finite cyclic group $\mathbb{G}$ of prime order p, generated by an element g.

The classical example of a key exchange protocol is the Diffie-Hellman key exchange [DH76], which is depicted in Figure 1. Let $\mathbb{G}$ be a cyclic group of prime order p, let g be a generator for $\mathbb{G}$, and let Alice and Bob be two users willing to establish a shared secret key. In the Diffie-Hellman key exchange protocol, Alice chooses a secret value x uniformly at random from $\mathbb{Z}_p$ and then sends the ephemeral public key value $X = g^x$ to Bob. Likewise, Bob chooses a secret value y in $\mathbb{Z}_p$ and sends $Y = g^y$ to Alice. Finally, after receiving the values X and Y, both Alice and Bob can compute a common secret $SK = g^{xy}$ using their corresponding secret values.

Despite its simplicity, the Diffie-Hellman key exchange protocol can be shown to be secure under reasonable assumptions with respect to adversaries which only eavesdrop on the communication without altering it. However, the same is not true with respect to active adversaries. The problem with it is that the Diffie-Hellman key exchange protocol does not include any form of authentication and can be easily compromised by an adversary that plays the role of one of the parties in the protocol. It thus becomes clear that to avoid such impersonation attacks and achieve security even in the presence of active adversaries, both parties need to somehow authenticate each other in addition to establishing a common secret.

Provable security. Even though we have already mentioned that a cryptographic scheme may possess several properties, such as data privacy and authenticity, we have not actually shown how one can prove that a cryptographic scheme has these properties. In the past, the most common approach to validate the security of a cryptographic scheme was to search for attacks and to declare a scheme secure if no attack is found that contradicts its security. Unfortunately, the problem with this approach is that we can never be certain that an attack does not exist. Hence, the security of the scheme can only be considered heuristic at best as the possibility that an attack exists cannot be excluded.

Another approach for proving the security of a cryptographic scheme, known as *provable security*, is to relate the security of a cryptographic scheme with that of its underlying primitives or computational problems. To achieve this goal, one needs to first specify the attacker's capabilities and the security goals that a given cryptographic scheme must meet. Next, one needs to specify the attacker's capabilities and the security goals for the underlying primitives and computational problems. Finally, one needs to provide a reduction which shows how to transform an adversary that breaks the security goals of a cryptographic scheme into an adversary against the security goals of the cryptographic primitives and computational problems on which the scheme is based.

One direct consequence of provable security is that it obviates the need to search for specific attacks against a cryptographic scheme. This is because as long as we are ready to believe that the underlying primitives are secure or the computational problems are hard, then there can be no attacks against the cryptographic scheme in question.

In order to illustrate how the provable security methodology can be used in practice, we also provide a concrete example on how to prove the security of a cryptographic scheme by considering the classical ElGamal public-key encryption scheme [ElG85]. But before doing so, we start by recalling some of the standard tools and computational problems that we use in this note.

2 Basic tools

In this section, we recall some of the definitions and basic tools that will be used in the remaining chapters, such as the syntax of code-based games and the description of bilinear maps.

2.1 Notation and conventions

Let $\mathbb{N}$ denote the set of natural numbers. If $n \in \mathbb{N}$, then $\{0,1\}^n$ denotes the set of n-bit strings, and $\{0,1\}^*$ is the set of all bit strings. The empty string is denoted ε. More generally, if S is a set, then S^n is the set of n-tuples of elements of S, $S^{\leq n}$ is the set of tuples of length at most n. If x is a string then $|x|$ denotes its length, and if S is a set then $|S|$ denotes its size. If S is finite, then $x \overset{R}{\leftarrow} S$ denotes the assignment to x of an element chosen uniformly at random from S. If $\mathcal{A}$ is an algorithm, then $y \leftarrow \mathcal{A}(x)$ denotes the assignment to y of the output of $\mathcal{A}$ on input x, and if $\mathcal{A}$ is randomized, then $y \overset{R}{\leftarrow} \mathcal{A}(x)$ denotes that the output of an execution of $\mathcal{A}(x)$ with fresh coins assigned to y. Unless otherwise indicated, an algorithm may be randomized. "PT" stands for polynomial time and "PTA" for polynomial-time algorithm or adversary. We denote by $k \in \mathbb{N}$ the security parameter. A function $\nu : \mathbb{N} \to [0,1]$ is said to be *negligible* if for every $c \in \mathbb{N}$ there exists a $k_c \in \mathbb{N}$ such that $\nu(k) \leq k^{-c}$ for all $k > k_c$, and it is said to be *overwhelming* if the function $|1 - \nu(k)|$ is negligible.

2.2 Code-based games

Some of the definitions in this note use code-based game-playing [BR06]. In such games, there exist procedures for initialization (**Initialize**) and finalization (**Finalize**) and procedures to respond to adversary oracle queries. A game G is executed with an adversary $\mathcal{A}$ as follows. First, **Initialize** executes and its outputs are the inputs to $\mathcal{A}$. Then $\mathcal{A}$ executes, its oracle queries being answered by the corresponding procedures of G. When $\mathcal{A}$ terminates, its output becomes the input to the **Finalize** procedure. The output of the latter, denoted $\mathrm{G}(\mathcal{A})$, is called the output of the game, and "$\mathrm{G}(\mathcal{A}) = y$" denotes the event that the output takes a value y. Boolean flags are assumed initialized to false. Games $\mathrm{G}_i, \mathrm{G}_j$ are *identical until* bad if their code differs only in statements that follow the setting of bad to true.

2.3 Represented groups

Let $\mathbb{G} = \langle g \rangle$ be a finite cyclic group of prime order p generated by an element g, where $k = |p|$ is the security parameter. Throughout this note, we will use the multiplicative notation for the group operation. Hence, g^0 denotes the identity element of $\mathbb{G}$ and g^u denotes the group element of $\mathbb{G}$ that results from multiplying u copies of g, for $u \in \mathbb{N}$. Note that $g^u = g^{u \bmod |\mathbb{G}|}$ by Lagrange's theorem.

Algorithms which operate on $\mathbb{G}$ will be given string representations of elements in $\mathbb{G}$. For that, we require an injective map $\_ : \mathbb{G} \to \{0,1\}^\ell$ associated to $\mathbb{G}$, where ℓ is the length of the representation of group elements. Similarly, when a number $i \in \mathbb{N}$ is an input to, or output of, an algorithm, it must be appropriately encoded, say in binary. We assume all necessary encoding methods are fixed, and do not normally write the $\_$ operators.

The schemes considered in this note are parameterized by a *group generator*, which is a PTA $\mathcal{G}$ that on input 1^k returns the description of a multiplicative group $\mathbb{G}$ of prime order p, where $2^k < p < 2^{k+1}$.

2.4 Bilinear maps

The pairing-based schemes that we consider are parameterized by a *pairing parameter generator*, which is a PTA $\mathcal{G}$ that on input 1^k returns the description of two multiplicative groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with an admissible map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $2^k < p < 2^{k+1}$. By admissible, we mean that the map is bilinear, non-degenerate, and efficiently computable. Bilinearity means that for all $a, b \in \mathbb{Z}_p^*$ and all $g \in \mathbb{G}$, we have $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$. By non-degenerate, we mean that $\hat{e}(g, g) = 1$ if and only if $g = 1$.

3 Standard complexity assumptions

CDH and DDH Problems. Two of the most common computational problems in finite cyclic groups are the computational Diffie-Hellman (CDH) and the decisional Diffie-Hellman (DDH) problems. In the CDH problem, the adversary is given a tuple (g, g^a, g^b) for random integers $a, b \in \mathbb{Z}_p^*$ and a random generator $g \overset{R}{\leftarrow} \mathbb{G}^*$ and its goal is to compute g^{ab}. In the DDH problem, the goal is to distinguish (g, g^a, g^b, g^{ab}) from (g, g^a, g^b, g^c) when g is a random generator for $\mathbb{G}$ and a, b, c are chosen uniformly at random from $\mathbb{Z}_p^*$.

To define more precisely the CDH problem with respect to a group generator $\mathcal{G}$, we consider the game $\mathbf{Exp}^{\mathrm{cdh}}_{\mathcal{G},k}(\mathcal{A})$ described in Figure 2 using the notation of code-based games. The game is defined by two procedures and is executed with an adversary $\mathcal{A}$ as follows. The procedure **Initialize** chooses a random generator $g \overset{R}{\leftarrow} \mathbb{G}^*$ and two random integers $x, y \overset{R}{\leftarrow} \mathbb{Z}_p^*$, computes $X \leftarrow g^x$ and $Y \leftarrow g^y$, and returns $((\mathbb{G}, p), g, X, Y)$ to $\mathcal{A}$. Eventually, the adversary ends the game by querying the **Finalize** procedure with a group element Z, which outputs true if $Z = g^{xy}$ and false, otherwise. The advantage $\mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G},k}(\mathcal{A})$ of an adversary $\mathcal{A}$ in solving the CDH problem relative to $\mathcal{G}$ is then defined as the probability that $\mathbf{Exp}^{\mathrm{cdh}}_{\mathcal{G},k}(\mathcal{A})$ outputs true. In other words,

$$\mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G},k}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\mathrm{cdh}}_{\mathcal{G},k}(\mathcal{A}) = \mathsf{true}\right].$$

In order to define more formally the DDH problem relative to a group generator $\mathcal{G}$, consider the games $\mathbf{Exp}^{\mathrm{ddh}\text{-}\beta}_{\mathcal{G},k}(\mathcal{A})$ described in Figure 2 for $\beta \in \{0,1\}$. Both games are defined by two procedures, which are executed with an adversary $\mathcal{A}$ as follows. In the game $\mathbf{Exp}^{\mathrm{ddh}\text{-}0}_{\mathcal{G},k}(\mathcal{A})$, the procedure **Initialize** chooses a random generator $g \overset{R}{\leftarrow} \mathbb{G}^*$ and random integers $x, y \overset{R}{\leftarrow} \mathbb{Z}_p^*$, sets

```
Game $\mathbf{Exp}^{\mathrm{cdh}}_{\mathcal{G},k}(\mathcal{A})$

proc Initialize($k$)
$(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$
$\overrightarrow{\mathbb{G}} \leftarrow (\mathbb{G}, p)$
$g \xleftarrow{R} \mathbb{G}^*$
$x \xleftarrow{R} \mathbb{Z}_p^*$ ; $X \leftarrow g^x$
$y \xleftarrow{R} \mathbb{Z}_p^*$ ; $Y \leftarrow g^y$
Return $(\overrightarrow{\mathbb{G}}, g, X, Y)$

proc Finalize($Z$)
Return $(Z = g^{xy})$
```

```
Game $\mathbf{Exp}^{\mathrm{ddh}\text{-}0}_{\mathcal{G},k}(\mathcal{A})$

proc Initialize($k$)
$(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$
$\overrightarrow{\mathbb{G}} \leftarrow (\mathbb{G}, p)$
$g \xleftarrow{R} \mathbb{G}^*$
$x \xleftarrow{R} \mathbb{Z}_p^*$ ; $X \leftarrow g^x$
$y \xleftarrow{R} \mathbb{Z}_p^*$ ; $Y \leftarrow g^y$
$z \leftarrow ab \mod p$ ; $Z \leftarrow g^z$
Return $(\overrightarrow{\mathbb{G}}, g, X, Y, Z)$

proc Finalize($\beta'$)
Return $(\beta' = 1)$
```

```
Game $\mathbf{Exp}^{\mathrm{ddh}\text{-}1}_{\mathcal{G},k}(\mathcal{A})$

proc Initialize($k$)
$(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$
$\overrightarrow{\mathbb{G}} \leftarrow (\mathbb{G}, p)$
$g \xleftarrow{R} \mathbb{G}^*$
$x \xleftarrow{R} \mathbb{Z}_p^*$ ; $X \leftarrow g^x$
$y \xleftarrow{R} \mathbb{Z}_p^*$ ; $Y \leftarrow g^y$
$z \xleftarrow{R} \mathbb{Z}_p^*$ ; $Z \leftarrow g^z$
Return $(\overrightarrow{\mathbb{G}}, g, X, Y, Z)$

proc Finalize($\beta'$)
Return $(\beta' = 1)$
```

Figure 2: Games $\mathbf{Exp}^{\mathrm{cdh}}_{\mathcal{G},k}(\mathcal{A})$, $\mathbf{Exp}^{\mathrm{ddh}\text{-}0}_{\mathcal{G},k}(\mathcal{A})$, and $\mathbf{Exp}^{\mathrm{ddh}\text{-}1}_{\mathcal{G},k}(\mathcal{A})$ defining the advantage of an adversary $\mathcal{A}$ against the CDH and DDH problems relative to a group generator $\mathcal{G}$ and security parameter k.

$z \leftarrow xy \mod p$, computes $X \leftarrow g^x$, $Y \leftarrow g^y$, and $Z \leftarrow g^z$, and returns $((\mathbb{G}, p), g, X, Y, Z)$ to $\mathcal{A}$. In the game $\mathbf{Exp}^{\mathrm{ddh}\text{-}1}_{\mathcal{G},k}(\mathcal{A})$, the procedure **Initialize** chooses a random generator $g \xleftarrow{R} \mathbb{G}^*$ and random integers $x, y, z \xleftarrow{R} \mathbb{Z}_p^*$, computes $X \leftarrow g^x$, $Y \leftarrow g^y$, and $Z \leftarrow g^z$, and returns $((\mathbb{G}, p), g, X, Y, Z)$ to $\mathcal{A}$. In both games, the adversary eventually ends the game by querying the **Finalize** procedure with a guess β, which in turn returns true if $\beta' = 1$ and false, otherwise. The advantage $\mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{G},k}(\mathcal{A})$ of an adversary $\mathcal{A}$ in solving the DDH problem relative to $\mathcal{G}$ is then defined as the probability that $\mathbf{Exp}^{\mathrm{ddh}\text{-}0}_{\mathcal{G},k}(\mathcal{A})$ outputs true minus the probability that $\mathbf{Exp}^{\mathrm{ddh}\text{-}1}_{\mathcal{G},k}(\mathcal{A})$ outputs true. That is,

$$\mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{G},k}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\mathrm{ddh}\text{-}0}_{\mathcal{G},k}(\mathcal{A}) = \mathsf{true}\right] - \Pr\left[\mathbf{Exp}^{\mathrm{ddh}\text{-}1}_{\mathcal{G},k}(\mathcal{A}) = \mathsf{true}\right].$$

Finally, the CDH and DDH problems are said to be hard relative to $\mathcal{G}$ if $\mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G},k}(\mathcal{A})$ and $\mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{G},k}(\mathcal{A})$ are negligible functions in k for all PTAs $\mathcal{A}$.

BDH and BDDH Problems. In the bilinear-map setting, two of the most common computational problems are the bilinear Diffie-Hellman (BDH) problem and its decisional version, the bilinear decisional Diffie-Hellman (BDDH) problem [BF01, Jou04]. While in the BDH problem, the goal is to compute $\hat{e}(g,g)^{abc}$ given a tuple (g, g^a, g^b, g^c) for random integers $a, b, c \in \mathbb{Z}_p^*$, in the BDDH problem, the goal is to distinguish the element $\hat{e}(g,g)^{abc}$ from a random element of $\mathbb{G}_T^*$.

More precisely, the advantage $\mathbf{Adv}^{\mathrm{bdh}}_{\mathcal{G},k}(\mathcal{A})$ of an adversary $\mathcal{A}$ in solving the BDH problem relative to a pairing parameter generator $\mathcal{G}$ is defined as the probability that the adversary $\mathcal{A}$ outputs $\hat{e}(g,g)^{abc}$ on input $((\mathbb{G}, \mathbb{G}_T, p, \hat{e}), g, g^a, g^b, g^c)$ for randomly chosen $g \xleftarrow{R} \mathbb{G}^*$ and $a, b, c \xleftarrow{R} \mathbb{Z}_p^*$. Using the code-based notation, $\mathbf{Adv}^{\mathrm{bdh}}_{\mathcal{G},k}(\mathcal{A})$ corresponds to probability that the game $\mathbf{Exp}^{\mathrm{bdh}}_{\mathcal{G},k}(\mathcal{A})$ returns true in Figure 3.

To define the advantage $\mathbf{Adv}^{\mathrm{bddh}}_{\mathcal{G},k}(\mathcal{A})$ of an adversary $\mathcal{A}$ in solving the BDDH problem relative to $\mathcal{G}$, we consider the game $\mathbf{Exp}^{\mathrm{bddh}\text{-}\beta}_{\mathcal{G},k}(\mathcal{A})$ between $\mathcal{A}$ and a challenger in Figure 3 for $\beta \in \{0, 1\}$. In these games, the procedure **Initialize** first chooses a random generator $g \xleftarrow{R} \mathbb{G}^*$, random integers $a, b, c \xleftarrow{R} \mathbb{Z}_p^*$, and a random element $T \xleftarrow{R} \mathbb{G}_T$. If $\beta = 1$, **Initialize**

```
Game $\mathbf{Exp}^{\mathrm{bdh}}_{\mathcal{G},k}(\mathcal{A})$

proc Initialize($k$)
$(\mathbb{G}, \mathbb{G}_T, p, \hat{e}) \overset{R}{\leftarrow} \mathcal{G}(1^k)$
$\overrightarrow{\mathbb{G}} \leftarrow (\mathbb{G}, \mathbb{G}_T, p, \hat{e})$
$g \overset{R}{\leftarrow} \mathbb{G}^*$
$a \overset{R}{\leftarrow} \mathbb{Z}_p^*$ ; $A \leftarrow g^a$
$b \overset{R}{\leftarrow} \mathbb{Z}_p^*$ ; $B \leftarrow g^b$
$c \overset{R}{\leftarrow} \mathbb{Z}_p^*$ ; $C \leftarrow g^b$
Return $(\overrightarrow{\mathbb{G}}, g, A, B, C)$

proc Finalize($Z$)
Return $(Z = \hat{e}(g,g)^{abc})$
```

```
Game $\mathbf{Exp}^{\mathrm{bddh}\text{-}0}_{\mathcal{G},k}(\mathcal{A})$

proc Initialize($k$)
$(\mathbb{G}, \mathbb{G}_T, p, \hat{e}) \overset{R}{\leftarrow} \mathcal{G}(1^k)$
$\overrightarrow{\mathbb{G}} \leftarrow (\mathbb{G}, \mathbb{G}_T, p, \hat{e})$
$g \overset{R}{\leftarrow} \mathbb{G}^*$
$a \overset{R}{\leftarrow} \mathbb{Z}_p^*$ ; $A \leftarrow g^a$
$b \overset{R}{\leftarrow} \mathbb{Z}_p^*$ ; $B \leftarrow g^b$
$c \overset{R}{\leftarrow} \mathbb{Z}_p^*$ ; $C \leftarrow g^b$
$z \leftarrow abc \mod p$ ; $Z \leftarrow \hat{e}(g,g)^z$
Return $(\overrightarrow{\mathbb{G}}, g, A, B, C, Z)$

proc Finalize($\beta'$)
Return $(\beta' = 1)$
```

```
Game $\mathbf{Exp}^{\mathrm{bddh}\text{-}1}_{\mathcal{G},k}(\mathcal{A})$

proc Initialize($k$)
$(\mathbb{G}, \mathbb{G}_T, p, \hat{e}) \overset{R}{\leftarrow} \mathcal{G}(1^k)$
$\overrightarrow{\mathbb{G}} \leftarrow (\mathbb{G}, \mathbb{G}_T, p, \hat{e})$
$g \overset{R}{\leftarrow} \mathbb{G}^*$
$a \overset{R}{\leftarrow} \mathbb{Z}_p^*$ ; $A \leftarrow g^a$
$b \overset{R}{\leftarrow} \mathbb{Z}_p^*$ ; $B \leftarrow g^b$
$c \overset{R}{\leftarrow} \mathbb{Z}_p^*$ ; $C \leftarrow g^b$
$z \overset{R}{\leftarrow} \mathbb{Z}_p^*$ ; $Z \leftarrow \hat{e}(g,g)^z$
Return $(\overrightarrow{\mathbb{G}}, g, A, B, C, Z)$

proc Finalize($\beta'$)
Return $(\beta' = 1)$
```

Figure 3: Games $\mathbf{Exp}^{\mathrm{bdh}}_{\mathcal{G},k}(\mathcal{A})$, $\mathbf{Exp}^{\mathrm{bddh}\text{-}0}_{\mathcal{G},k}(\mathcal{A})$, and $\mathbf{Exp}^{\mathrm{bddh}\text{-}1}_{\mathcal{G},k}(\mathcal{A})$ defining the advantage of an adversary $\mathcal{A}$ against the BDH and BDDH problems relative to a pairing parameter generator $\mathcal{G}$ and security parameter k.

returns the tuple $((\mathbb{G}, \mathbb{G}_T, p, \hat{e}), g, g^a, g^b, g^c, \hat{e}(g,g)^{abc})$ to $\mathcal{A}$; if $\beta = 0$, it returns the tuple $((\mathbb{G}, \mathbb{G}_T, p, \hat{e}), g, g^a, g^b, g^c, T)$ instead. The advantage $\mathbf{Adv}^{\mathrm{bddh}}_{\mathcal{G},k}(\mathcal{A})$ is then defined as the probability that game $\mathbf{Exp}^{\mathrm{bddh}\text{-}0}_{\mathcal{G},k}(\mathcal{A})$ outputs true minus the probability that game $\mathbf{Exp}^{\mathrm{bddh}\text{-}1}_{\mathcal{G},k}(\mathcal{A})$ outputs true. Finally, the BDH and BDDH problems are said to be hard relative to $\mathcal{G}$ if $\mathbf{Adv}^{\mathrm{bdh}}_{\mathcal{G},k}(\mathcal{A})$ and $\mathbf{Adv}^{\mathrm{bddh}}_{\mathcal{G},k}(\mathcal{A})$ are negligible functions in k for all PTAs $\mathcal{A}$.

4 Example: Public-key encryption

In order to illustrate how provable security can be used in practice, we provide in this section a proof of security for the classical ElGamal public-key encryption scheme [ElG85] as an example. Towards this goal, we start by recalling the formal definition of a public-key encryption scheme and what it means to be secure. Next, we describe the ElGamal public-key encryption scheme and show that it meets the notion of indistinguishability under chosen-plaintext attacks under the decisional Diffie-Hellman assumption described in Section 3.

Syntax. A *public-key encryption* scheme (PKE) is defined by a tuple of algorithms $\mathsf{PKE} = (\mathsf{PG}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ and a message space $\mathcal{M}$, providing the following functionality. Via $pars \overset{R}{\leftarrow} \mathsf{PG}(1^k)$, one can run the probabilistic parameter generation algorithm PG to setup the common parameter *pars* for a given security parameter k. Via $(pk, sk) \overset{R}{\leftarrow} \mathsf{KeyGen}(pars)$, a user can run the probabilistic algorithm KeyGen to obtain a pair (pk, sk) of public and secret keys with respect to common parameter *pars*. Via $C \overset{R}{\leftarrow} \mathsf{Enc}(pk, m)$, one can send an encrypted message $m \in \mathcal{M}$ to the user with public pk. Finally, via $m \leftarrow \mathsf{Dec}(sk, C)$, the user in possession of the secret key sk and a ciphertext C can run the deterministic decryption algorithm to recover the underlying plaintext m. For correctness, it is required that for all honestly generated keys $(pk, sk) \overset{R}{\leftarrow} \mathsf{KeyGen}$, for all messages $m \in \mathcal{M}$, $m = \mathsf{Dec}(sk, \mathsf{Enc}(pk, m))$ holds with all but negligible probability.

Security definition. The now-standard definition of security of PKE schemes, suggested by Goldwasser and Micali [GM84], is indistinguishability under *chosen-plaintext attacks* (IND-CPA).

```
Game $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{PKE},k}(\mathcal{A})$

proc Initialize($k$)
    $pars \xleftarrow{R} \mathsf{PG}(1^k)$
    $(pk, sk) \xleftarrow{R} \mathsf{KeyGen}(pars)$
    Return $pk$

proc LR($m_0^*, m_1^*$)
    $C^* \xleftarrow{R} \mathsf{Enc}(pk, m_\beta^*)$
    Return $C^*$

proc Finalize($\beta'$)
    Return $(\beta' = 1)$
```

Figure 4: Game $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{PKE},k}(\mathcal{A})$ for $\beta \in \{0,1\}$ defining the IND-CPA security of a public-key encryption scheme $\mathsf{PKE} = (\mathsf{PG}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

In this security model, the adversary receives the public key of the scheme that he is trying to attack and his goal is to find a pair of messages of the same length whose encryptions he is able to distinguish. Since the adversary is allowed to choose the challenge messages after seeing the public key, this security notion implicitly provides security against key recovery attacks.

The precise definition of IND-CPA security considers the game $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{PKE},k}(\mathcal{A})$ described in Figure 4. $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{PKE},k}(\mathcal{A})$ contains three procedures, which are executed with an adversary $\mathcal{A}$ as follows. The procedure **Initialize** generates the common parameter $pars \xleftarrow{R} \mathsf{PG}(1^k)$ and a pair of public and secret keys $(pk, sk) \xleftarrow{R} \mathsf{KeyGen}(pars)$ and returns pk to $\mathcal{A}$. During the execution of the game, the adversary is allowed to make a *single* query (m_0^*, m_1^*) to the **LR** procedure, where $m_0^*, m_1^* \in \{0,1\}^*$ are assumed to have the same length. To answer it, the game $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{PKE},k}(\mathcal{A})$ generates a challenge ciphertext $C^* \xleftarrow{R} \mathsf{Enc}(pk, m_\beta^*)$ and gives C^* to $\mathcal{A}$. Eventually, the adversary ends the game by querying the **Finalize** procedure with a guess β' for the bit β used to generate the challenge ciphertext. The advantage $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},k}(\mathcal{A})$ of the adversary $\mathcal{A}$ in breaking the IND-CPA security of PKE is then defined as the probability that game $\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathsf{PKE},k}(\mathcal{A})$ outputs true minus the probability that game $\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathsf{PKE},k}(\mathcal{A})$ outputs true. Finally, we say that PKE is secure if $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},k}(\mathcal{A})$ is a negligible function in k for all PTAs $\mathcal{A}$.

ElGamal encryption. The ElGamal public-key encryption scheme, described in Figure 5, was proposed in [ElG85]. It can be seen as an adaptation of the Diffie-Hellman key exchange described in Figure 1 on page 3 to the public-key setting by fixing the first message sent by one the parties as the public key for that party.

As in the Diffie-Hellman key exchange, the ElGamal encryption scheme works over a finite cyclic group $\mathbb{G}$ of prime order p obtained via a group generator $\mathcal{G}$. To generate a pair (pk, sk) of public and secret keys, the user chooses a generator g for $\mathbb{G}$ and a random element $x \in \mathbb{Z}_p^*$, computes $X \leftarrow g^x$, and sets $pk = (\mathbb{G}, p, g, X)$ and $sk = (\mathbb{G}, p, g, x)$. To encrypt a message $m \in \mathbb{G}$ to a user with public key $pk = (\mathbb{G}, p, g, X)$, the sender simply chooses a random element $r \in \mathbb{Z}_p^*$ and outputs $(C_1, C_2) = (g^r, m \cdot X^r)$ as the ciphertext. To decrypt it, the user in possession of the secret key $sk = (\mathbb{G}, p, g, x)$ corresponding to $pk = (\mathbb{G}, p, g, X)$ computes $C_2 / {C_1}^x$ to recover the underlying message.

Security of ElGamal encryption. In order to prove that the ElGamal encryption scheme meets the IND-CPA security notion depicted in Figure 4 if the DDH problem is hard with respect to group generator $\mathcal{G}$, we will provide a reduction which relates the advantage of the adversary in breaking IND-CPA security game to the advantage of another adversary in breaking the DDH problem with respect to $\mathcal{G}$. More precisely, we want to prove the following theorem.

Theorem 4.1 Let EG refer to the ElGamal PKE scheme in Figure 5, let $\mathcal{G}$ be the underlying

```
PG(1^k):
    $(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$
    $pars \leftarrow (\mathbb{G}, p)$
    Return $pars$
```

```
KeyGen(pars):
    parse $pars$ as $(\mathbb{G}, p)$
    $g \xleftarrow{R} \mathbb{G}^*$
    $x \xleftarrow{R} \mathbb{Z}_p^*$ ; $X \leftarrow g^x$
    $sk \leftarrow (\mathbb{G}, p, g, x)$
    $pk \leftarrow (\mathbb{G}, p, g, X)$
    Return $(pk, sk)$
```

```
Enc(pk, m):
    $r \xleftarrow{R} \mathbb{Z}_p^*$ ; $C_1 \leftarrow g^r$
    $K \leftarrow X^r$
    $C_2 \leftarrow m \cdot K$
    Return $(C_1, C_2)$
```

```
Dec(sk, C):
    parse $C$ as $(C_1, C_2)$
    parse $sk$ as $(\mathbb{G}, p, g, x)$
    $m' \leftarrow C_2 / {C_1}^x$
    Return $m'$
```

Figure 5: The ElGamal public-key encryption scheme [ElG85], where $\mathcal{G}$ is a group generator.

group generator, let k be the security parameter, and let $\mathcal{A}$ be an adversary against IND-CPA security notion as depicted in Figure 4, making at most a single query to the **LR** procedure. Then, there exists an adversary $\mathcal{B}$ against the DDH problem with respect to $\mathcal{G}$, whose running time is that of $\mathcal{A}$ and such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{EG},k}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\text{ddh}}_{\mathcal{G},k}(\mathcal{B}).$$

The actual proof of Theorem 4.1 is quite simple and uses the fact that, in order to distinguish between the encryption of the challenge messages, the adversary needs to somehow distinguish the value $K = X^r = g^{rx}$ used to hide the message in C_2 from a random element in the group, when given the public key $pk = (\mathbb{G}, p, g, X = g^x)$ and the first part of the ciphertext, $C_1 = g^r$. However, to make this intuition more precise and to illustrate the usefulness of games in security proofs, we will prove Theorem 4.1 using a sequence of hybrid games.

Our proof contains a total of 5 games, which are described in Figure 6, in which games G_0 and G_4 correspond respectively to the games $\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathsf{EG},k}(\mathcal{A})$ and $\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathsf{EG},k}(\mathcal{A})$ of the IND-CPA security definition. Hence, in order to show that the advantage $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{EG},k}(\mathcal{A})$ of $\mathcal{A}$ against EG is negligible for all PTAs $\mathcal{A}$, it suffices to show that the probability that $\mathrm{G}_i^{\mathcal{A}}$ outputs true for $i = 0, \ldots, 4$ does not change significantly.

Proof: Consider the sequence of games depicted in Figure 6. By substituting the description of the ElGamal PKE scheme (EG) in Game $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{EG},k}(\mathcal{A})$ for $\beta \in \{0,1\}$ in Figure 4, we have that

$$\begin{aligned}\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{EG},k}(\mathcal{A}) &= \Pr\left[\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathsf{EG},k}(\mathcal{A}) = \mathsf{true}\right] - \Pr\left[\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathsf{EG},k}(\mathcal{A}) = \mathsf{true}\right] \\ &= \Pr\left[\,\mathrm{G}_0(\mathcal{A}) = \mathsf{true}\,\right] - \Pr\left[\,\mathrm{G}_4(\mathcal{A}) = \mathsf{true}\,\right]. \qquad (1)\end{aligned}$$

We now claim that there exists an adversary $\mathcal{B}_1$ against the DDH problem relative to $\mathcal{G}$ such that $\Pr[\,\mathrm{G}_0(\mathcal{A}) = \mathsf{true}\,] - \Pr[\,\mathrm{G}_1(\mathcal{A}) = \mathsf{true}\,] \leq \mathbf{Adv}^{\text{ddh}}_{\mathcal{G},k}(\mathcal{B})$. In order to prove this claim, we build $\mathcal{B}_1$ as follows. Let $((\mathbb{G}, p), g, X, Y, Z)$ be the input that $\mathcal{B}$ receives from the **Initialize** procedure in $\mathbf{Exp}^{\text{ddh-}\beta}_{\mathcal{G},k}(\mathcal{B}_1)$. $\mathcal{B}_1$ then sets $(\mathbb{G}, g)$ as the underlying group for EG and returns $pk = (\mathbb{G}, p, g, X)$ to $\mathcal{A}$ as the output of its **Initialize** procedure of $\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathsf{EG},k}(\mathcal{A})$. When $\mathcal{A}$ queries its **LR** procedure with a pair of messages (m_0^*, m_1^*), $\mathcal{B}_1$ simulates its behavior by setting $(C_1^*, C_2^*) = (Y, m_0^* \cdot Z)$ and returning it to $\mathcal{A}$. Finally, when $\mathcal{A}$ queries its **Finalize** procedure with a guess bit β', $\mathcal{B}_1$ queries its own **Finalize** procedure with β'.

Game G_0	**Game** G_1	**Game** G_2	**Game** G_3	**Game** G_4
proc Initialize(k)	**proc Initialize**(k)	**proc Initialize**(k)	**proc Initialize**(k)	**proc Initialize**(k)
$(\mathbb{G}, p) \overset{R}{\leftarrow} \mathcal{G}(1^k)$	$(\mathbb{G}, p) \overset{R}{\leftarrow} \mathcal{G}(1^k)$	$(\mathbb{G}, p) \overset{R}{\leftarrow} \mathcal{G}(1^k)$	$(\mathbb{G}, p) \overset{R}{\leftarrow} \mathcal{G}(1^k)$	$(\mathbb{G}, p) \overset{R}{\leftarrow} \mathcal{G}(1^k)$
$g \overset{R}{\leftarrow} \mathbb{G}^*$	$g \overset{R}{\leftarrow} \mathbb{G}^*$	$g \overset{R}{\leftarrow} \mathbb{G}^*$	$g \overset{R}{\leftarrow} \mathbb{G}^*$	$g \overset{R}{\leftarrow} \mathbb{G}^*$
$x \overset{R}{\leftarrow} \mathbb{Z}_p^*\ ;\ X \leftarrow g^x$	$x \overset{R}{\leftarrow} \mathbb{Z}_p^*\ ;\ X \leftarrow g^x$	$x \overset{R}{\leftarrow} \mathbb{Z}_p^*\ ;\ X \leftarrow g^x$	$x \overset{R}{\leftarrow} \mathbb{Z}_p^*\ ;\ X \leftarrow g^x$	$x \overset{R}{\leftarrow} \mathbb{Z}_p^*\ ;\ X \leftarrow g^x$
$sk \leftarrow (\mathbb{G}, p, g, x)$	$sk \leftarrow (\mathbb{G}, p, g, x)$	$sk \leftarrow (\mathbb{G}, p, g, x)$	$sk \leftarrow (\mathbb{G}, p, g, x)$	$sk \leftarrow (\mathbb{G}, p, g, x)$
$pk \leftarrow (\mathbb{G}, p, g, X)$	$pk \leftarrow (\mathbb{G}, p, g, X)$	$pk \leftarrow (\mathbb{G}, p, g, X)$	$pk \leftarrow (\mathbb{G}, p, g, X)$	$pk \leftarrow (\mathbb{G}, p, g, X)$
Return pk	Return pk	Return pk	Return pk	Return pk
proc LR(m_0^*, m_1^*)	**proc LR**(m_0^*, m_1^*)	**proc LR**(m_0^*, m_1^*)	**proc LR**(m_0^*, m_1^*)	**proc LR**(m_0^*, m_1^*)
$r \overset{R}{\leftarrow} \mathbb{Z}_p^*\ ;\ C_1^* \leftarrow g^r$	$r \overset{R}{\leftarrow} \mathbb{Z}_p^*\ ;\ C_1^* \leftarrow g^r$	$r \overset{R}{\leftarrow} \mathbb{Z}_p^*\ ;\ C_1^* \leftarrow g^r$	$r \overset{R}{\leftarrow} \mathbb{Z}_p^*\ ;\ C_1^* \leftarrow g^r$	$r \overset{R}{\leftarrow} \mathbb{Z}_p^*\ ;\ C_1^* \leftarrow g^r$
$K \leftarrow X^r$	$\boxed{K \overset{R}{\leftarrow} \mathbb{G}^*}$	$K \overset{R}{\leftarrow} \mathbb{G}^*$	$K \overset{R}{\leftarrow} \mathbb{G}^*$	$\boxed{K \leftarrow X^r}$
$C_2^* \leftarrow m_0^* \cdot K$	$C_2^* \leftarrow m_0^* \cdot K$	$\boxed{C_2^* \overset{R}{\leftarrow} \mathbb{G}}$	$\boxed{C_2^* \leftarrow m_1^* \cdot K}$	$C_2^* \leftarrow m_1^* \cdot K$
Return (C_1^*, C_2^*)	Return (C_1^*, C_2^*)	Return (C_1^*, C_2^*)	Return (C_1^*, C_2^*)	Return (C_1^*, C_2^*)
proc Finalize(β')	**proc Finalize**(β')	**proc Finalize**(β')	**proc Finalize**(β')	**proc Finalize**(β')
Return $(\beta' = 1)$	Return $(\beta' = 1)$	Return $(\beta' = 1)$	Return $(\beta' = 1)$	Return $(\beta' = 1)$

Figure 6: Sequence of games for the security proof of the ElGamal PKE scheme described in Figure 5, where the rectangular boxes indicate differences with respect to the previous game. $\mathcal{G}$ is the underlying group generator used in the ElGamal PKE scheme.

Given the above description of $\mathcal{B}_1$, it is not hard to see that, when we execute $\mathcal{B}_1$ in $\mathbf{Exp}^{\text{ddh-0}}_{\mathcal{G},k}(\mathcal{B}_1)$, $\mathcal{B}_1$ simulates $\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathsf{EG},k}(\mathcal{A})$ to $\mathcal{A}$. Likewise, when $\mathcal{B}_1$ interacts with the challenger in $\mathbf{Exp}^{\text{ddh-1}}_{\mathcal{G},k}(\mathcal{B}_1)$, it simulates $\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathsf{EG},k}(\mathcal{A})$ to $\mathcal{A}$. Thus,

$$\Pr\left[\,\mathrm{G}_0(\mathcal{A}) = \mathsf{true}\,\right] \;=\; \Pr\left[\,\mathbf{Exp}^{\text{ddh-0}}_{\mathcal{G},k}(\mathcal{B}_1) = \mathsf{true}\,\right]$$
$$\Pr\left[\,\mathrm{G}_1(\mathcal{A}) = \mathsf{true}\,\right] \;=\; \Pr\left[\,\mathbf{Exp}^{\text{ddh-1}}_{\mathcal{G},k}(\mathcal{B}_1) = \mathsf{true}\,\right]$$

which implies that

$$\Pr\left[\,\mathrm{G}_0(\mathcal{A}) = \mathsf{true}\,\right] - \Pr\left[\,\mathrm{G}_1(\mathcal{A}) = \mathsf{true}\,\right] \;\leq\; \mathbf{Adv}^{\text{ddh}}_{\mathcal{G},k}(\mathcal{B}_1)\ . \tag{2}$$

Due to the similarities between games $\mathrm{G}_0(\mathcal{A})$ and $\mathrm{G}_4(\mathcal{A})$ and games $\mathrm{G}_1(\mathcal{A})$ and $\mathrm{G}_3(\mathcal{A})$, we can easily build an adversary $\mathcal{B}_2$ against the DDH problem relative to $\mathcal{G}$ such that

$$\Pr\left[\,\mathrm{G}_3(\mathcal{A}) = \mathsf{true}\,\right] - \Pr\left[\,\mathrm{G}_4(\mathcal{A}) = \mathsf{true}\,\right] \;\leq\; \mathbf{Adv}^{\text{ddh}}_{\mathcal{G},k}(\mathcal{B}_2)\ . \tag{3}$$

Finally, we note that the differences between games $\mathrm{G}_1(\mathcal{A})$ and $\mathrm{G}_2(\mathcal{A})$ and between games $\mathrm{G}_2(\mathcal{A})$ and $\mathrm{G}_3(\mathcal{A})$ are purely syntactic since, in all of these cases, C_2^* is a random element in the group which does not depend on the guess bit β used to select the challenge message. Hence,

$$\Pr\left[\,\mathrm{G}_1(\mathcal{A}) = \mathsf{true}\,\right] = \Pr\left[\,\mathrm{G}_2(\mathcal{A}) = \mathsf{true}\,\right] = \Pr\left[\,\mathrm{G}_3(\mathcal{A}) = \mathsf{true}\,\right]\ . \tag{4}$$

The proof of Theorem 4.1 follows by combining equations 1, 2, 3, and 4 and by noticing that the adversary $\mathcal{B}$ of the theorem statement runs $\mathcal{B}_1$ with probability 1/2 and $\mathcal{B}_2$ with probability 1/2. ▌

References

[Bel11] Mihir Bellare. CSE 207 – modern cryptography. Lecture Notes, 2011. `http://www-cse.ucsd.edu/users/mihir/cse207`. (Cited on page 1.)

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany. (Cited on page 6.)

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Germany. (Cited on page 4.)

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. (Cited on pages 2 and 3.)

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985. (Cited on pages 4, 7, 8, and 9.)

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. (Cited on page 7.)

[Jou04] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, September 2004. (Cited on page 6.)

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949. (Cited on page 1.)