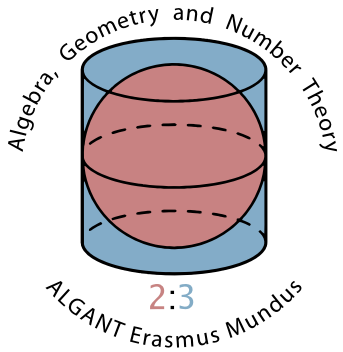
Angela Zottarel

Encryption from Weaker Assumptions



Master thesis defended on June 28, 2010.
Thesis Advisor: dr. Eike Kiltz.

Matematisch Instituut
Universiteit Leiden

Università degli Studi di Padova
Facoltà di Scienze MM.FF.NN

Encryption from Weaker Assumptions

Angela Zottarel

Supervisor: Eike Kiltz

Contents

Acknowledgments

I first thank my advisor Eike Kiltz for patiently guiding me through the experience of writing a master thesis.

Cảm ơn Dung, terima kasih Novi and xiè xiè Liu, for all the time spent together and for introducing me to the world of chili and ginger. I will never forget our ritual Saturday dinner. I thank all my friends in Padova; especially Liviana, for her cakes, her sweetness and for her different view about fishing-frogs. I thank Alberto Vezzani for visiting Nederland with us, also with the worst weather conditions.

I thank my family: grazie per avermi sempre sostenuto nelle mie scelte e per aver creduto in me. Quest'anno non siete andati in Canada, quindi niente scuse!

I thank my little notebook, for showing the first signs of breakdown only on the very last day.

Last, but not least, I thank Valerio: sharing my life with you means so much to me. Without your infinite patience and without your help I would not be able to finish this thesis.

Introduction

Concepts of Provable Security

In the early days of cryptography, security cryptographic primitives relied only on designer's fantasy and personal skill and they were commonly considered "secure" until someone found a way to break them. In the late 20th century, cryptography's world changed: intuitive ideas about security were formalized in a consistent science. In particular, a new way of formalizing security was developed, called *provable security*. In this work we will mainly deal with a particular area of cryptography, the one studying the design and analysis of public-key encryption schemes. An encryption scheme is a tuple of algorithms used to ensure secrecy while exchanging messages over a public channel. Provable security in our setting consists of three steps: the first one is giving a rigorous definition of security, the second one is giving the construction of a scheme, the last one is giving a rigorous mathematical proof showing that the scheme satisfies the security definition given in the first stage (unless one can solve a problem considered to be hard, such as discrete logarithm). We will explain these steps in more details in the following sections.

Security Definitions. A secure encryption scheme is often thought as a "secure envelope" (containing a message or *plaintext*) which can be opened, and so read, only by the addressee [Sho98]. The first formal definition of security, known as *semantic security* [GM84], formalizes the intuition that a ciphertext does not leak any information about the message. Anyway, another property has to be taken into account in order to create a secure envelope. The crucial point is that in a sealed envelope no one can meaningfully modify the plaintext. This property is called non malleability: we do not want that an attacker can change the ciphertext and obtain a new plaintext related to the old one. Again, a secure envelope is non malleable, but it has been shown that semantic security (in general) does not assure non malleability [BDJR98].
It turns out that *chosen ciphertext security* is a stronger notion than non malleability [BDJR98] and nowadays it is believed to be the "right" notion of security. Security against chosen-ciphertext attack (CCA) is formalized by letting the adversary interact with the honest party that decrypts any ciphertext (i.e., any bit string) the adversary sends [NY90,RS92]. Even if it is clear that this kind of security is needed to simulate a secure envelope, it took a while before people understood the importance of chosen-ciphertext security, and still now someone thinks that the usage of CCA-secure scheme is "unjustified". In reality, no one of us expects that an honest party voluntarily decrypts any arbitrary ciphertext sent by an adversary. Then, we are pushed to believe that chosen ciphertext attacks are not realistic in everyday life. Nevertheless, past attacks, as the one from Bleicherbacher [Ble98] against the RSA standard PKCS#1, showed that we are wrong, and it may be the case that such a scenario is not so far from being actual. In fact, an adversary may be unable to obtain a complete decryption of its ciphertext, but it could as well learn some partial information

about the plaintext looking at the reaction of the receiver. For example, consider the situation in which A interacts with her broker B and an adversary E knows that a ciphertext c could mean either "*sell shares*" or "*buy shares*". At this point, E can send the ciphertext c to B on the behalf of A and looks at the reaction of B. It is very likely that E can deduce the plaintext from the broker reactions.
Even if this shows that chosen ciphertext attacks are indeed practical, someone still prefers less secure cryptosystems because provable CCA security can be very expensive in terms of efficiency and practicality. Anyway, this is not longer an issue in the light of the new practical CCA secure schemes, whose first example was given by Cramer and Shoup in [CS98].

Public-Key Encryption Schemes. A public-key encryption scheme consists in three efficient algorithms. The first one takes a security parameter in $\mathbb{N}$ as input and returns two keys: a secret key that will be kept by the one that runs the algorithm and a public key that will be given to everyone who wants to communicate with him. The second algorithm takes as input a message and a public key and outputs the ciphertext that will be given to the addressee. The last algorithm, upon input a ciphertext and a secret key, outputs the decryption of the ciphertext. Once we describe these three algorithms, an encryption scheme is fully specified.

Proofs by Reduction. The modern approach to prove security consists in relating a way to break the scheme to a solution of a problem that is assumed to be hard (for example, factoring a large number that is a product of two primes). A proof by reduction works as follows: we take an arbitrary "efficient" algorithm $\mathcal{A}$ (i.e., the adversary) that tries to break our scheme. Then, we show that, using $\mathcal{A}$, we can build another efficient algorithm $\mathcal{B}$ that solves a hard problem. The key point is that as long as $\mathcal{A}$ succeeds in breaking the scheme, the algorithm $\mathcal{B}$ is able to solve the problem. Since we are assuming that the problem is hard, we have that algorithm $\mathcal{A}$ breaks the scheme just with a low probability. We point out that such a proof must hold for *every* efficient algorithm trying to break the scheme, and not just for some special ones.
In a proof by reduction there is an important feature to keep in mind. For instance, consider a reduction showing that breaking a scheme in time t and with probability ε implies solving a problem in time $t+\omega_1$ with probability ε/ω_2. It is clear that the two factors ω_1 and ω_2 are of the uttermost importance. If, for example, the factor ω_1 was exponential, we would have proved that breaking our scheme corresponds to solve a given problem in exponential time, which does not fit the purpose. This is one of the reasons why tightness was introduced. Informally, a tight reduction is a reduction in which the ratio between the success probability and the running time of an algorithm solving a given problem is "close" to the ratio between the success probability and the running time of an algorithm breaking the scheme. Tight reductions are not only a theoretical issue in cryptography, but they have an important practical consequence: loose reductions mean loss of security, thus large keys are needed to gain a satisfactory level of security (that can be achieved with sensibly smaller keys if the reduction is tight). It is often the case that, due to a loose reduction, a scheme requires keys that are totally impractical. As an example, consider that an exponentiation in a group where the elements are represented by r bit strings requires $\mathcal{O}(r^3)$ time. Hence, an exponentiation in 160-bit group is 4 times faster than an exponentiation in 256-bit group [BR09]. This shows that the key-size heavily affects the efficiency of a scheme.

Idealizations in Cryptography

Sometimes, in cryptography, some components of a scheme or some components of a proof model are idealized. There are several reasons why we decide to use models that differ from reality. The first one is that the proofs sometimes require tools that are not yet developed; another one is that it is possible to give easier proofs. In this work, we use two particular types of idealized model: the random oracle model and the generic group model.

Random Oracle Model. In the first part of this thesis, we will work in the random oracle model [BR93]. This model applies to cryptographic schemes that make use of a hash function. In the security proof, the hash functions are modeled as truly random ones. Since in the real world no truly random function exists, it is clear the this model offers an idealized environment for our proofs. This means that even if a scheme is secure in the random oracle model, we do not know anything about its security in the real world. This is because when we implement a scheme, we have to replace the random function by a efficiently computable function (such as SHA-1), and so we lose the properties given by the random oracle. In fact, there are examples of schemes that are provably secure in the random oracle model, but that result completely insecure when *any* efficiently computable function is used in place of a truly random one [CGH04]. At this point, one can wonder why the random oracle is used at all. The first reason is that the security in the random oracle model is achieved with schemes that are more efficient than schemes proved secure in the standard model. Anyway, this alone cannot be a valid reason. The point is that schemes proved to be secure in the random oracle model are widely used nowadays and still there are only few real attacks that break these schemes.

Generic Group Model. Proofs by reductions are based on the assumption that an underlying problem is not easy to solve. Thus, it is important to study the hardness of problems to show that it is meaningful to base a scheme on them. Anyway, with our current knowledge, no unconditional proof of hardness exists. This is due to the fact that we need results in complexity theory still far from being reached. As an example, one may consider that any unconditional proof of hardness would imply a proof that $\mathcal{P} \neq \mathcal{N}P$. To overcome this problem, Shoup proposed a restricted computational model in which no such deep results are required. The generic group [Sho97] is a model where the adversary does not use any specific property of a given group but he can perform just basic group operations (multiplications, inversions). For example, Shoup proved that the computational Diffie-Hellman (CDH) problem is unconditionally hard in the generic group model. Of course, a proof of hardness in generic groups does not mean that a problem is hard in the real world. The point is that, in the real world, the adversary can use any further property of the group that may allow him to be faster in solving the problem. Moreover, the generic group model has the same weakness found in the random oracle model: it is possible to prove that some problems are hard in generic groups but they are trivial to solve, as soon as we are in the real world [De02].

Our Results

In this thesis we will present a CCA secure encryption scheme as main result. Its security is based on an assumption that is equivalent to the CDH assumption in the asymptotic sense, but weaker in the concrete setting (generically).
The assumptions that we will use in this thesis are related to the discrete logarithm assumption. Fix a cyclic group $\mathbb{G}$ of order a prime number q and a generator g of $\mathbb{G}$. Given g^x in $\mathbb{G}$, for a uniform x in $\mathbb{Z}_q$, the discrete logarithm problem consists in finding $x \bmod q$.

The computational Diffie-Hellman problem is closely related to this problem: given g, g^x, g^y in $\mathbb{G}$, computing g^{xy} is the computational Diffie-Hellman problem. The CDH assumption states that solving the CDH problem is computationally infeasible.
We will start introducing a generalization of the CDH problem. We will consider the decisional linear assumption given in [BBS04] and we will present the computational version, which we call (computational) linear assumption. It is well known [Sha07] that the linear assumption is asymptotically equivalent to the CDH assumption. Anyway this is not a tight equivalence generically and our main technical result is proving in generic groups that solving the linear assumption requires solving two instances of the CDH problem. With other words, with just one arbitrary call to a CDH oracle is hard to solve the linear problem in generic groups.
We generalize the linear problem to the n-linear problem, which can be also seen as the computational version of the decisional n-linear problem [Sha07,HK07]. We will give the construction of a scheme that bases its security against chosen-plaintext attack on the linear assumptions, but as for the Hashed Elgamal encryption scheme [ABR01], security against chosen ciphertext attacks requires more than just the linear assumptions. More precisely, we will show that breaking our scheme with a chosen ciphertext attack is tightly equivalent to solving the strong linear assumption (which is the linear assumption with access to an oracle for the decisional linear assumption). Nevertheless, it would be preferable to have a scheme whose security is based on some more common assumption (like the linear assumption), to be sure that the problems we consider hard are indeed such. This is why we will give the construction of another scheme, using the twining technique presented in [CKS08]. This new scheme is secure against chosen ciphertext attack under an assumption similar to the strong linear one, but we will prove it is equivalent to the linear assumption. As a result, our scheme is CCA-secure under the linear assumption and, using the technical proof about the generic hardness of the linear assumption, it generically requires more than one CDH instance to be broken.
In the last part of our work, as an alternative to the decisional Diffie-Hellman assumption, we will define the decisional vector DH assumption. Similarly to the decisional linear assumption, it will turn out that our assumption is generically weaker than the DDH assumption. It is known that, as soon as a group has a efficiently computable bilinear map (we will give the definition in section 1.6.1), the decisional Diffie-Hellman problem is trivial [JN01]. This is not the case for our assumption; in fact, we will prove that in generic bilinear groups our problem is still hard. We will also present a scheme that is CPA-secure under the decisional vector DH assumption.

Related Work

The linear assumptions, in their decisional version, were introduced by Boneh, Boyen and Shacham in 2004 [BBS04]. They proved that this family of assumptions is weaker than the DDH assumption in the sense that the decisional linear assumptions hold in generic bilinear groups. They constructed a group signature that bases its security on the decisional linear assumptions. In 2007, Shacham [Sha07] proved that the linear assumptions are a family of progressively weaker assumptions, meaning that an algorithm solving the decisional n-linear problem in generic $(n-1)$-linear groups succeeds with negligible probability. He also showed that it is possible to use the decisional linear family to build a generalization of the Cramer-Shoup encryption scheme. The decisional linear assumptions were introduced also by Hofheinz and Kiltz in 2007 [HK07] in order to construct a key encapsulation mechanism; our n-linear assumption can be seen as the computational version of the decisional n-linear

assumption.
Cash, Kiltz and Shoup [CKS08] proposed the strong twin Diffie-Hellman problem and they built a "trapdoor test" that allows to answer decision oracle queries for the Diffie-Hellman problem without knowing the relative discrete logarithm. They constructed a variant of the hashed Elgamal encryption scheme whose CCA-security is based on the strong twin problem. Finally, they used their trapdoor lemma to prove that the strong twin assumption is equivalent to the CDH assumption, showing in this way that their scheme is CCA-secure under the CDH assumption. Our scheme can be viewed as a generalization of their work to the n-linear assumption.
In 2008 Huang et al [HYXZ08] proposed a new Diffie-Hellman problem exploiting the properties of the companion matrix of some irreducible polynomial. They showed that their new assumption is hard in generic bilinear groups, proving then that it is a weaker assumption then the DDH one. They also gave an encryption scheme based on their new assumption. Our assumption can be seen as a specialization of theirs, in the case when the polynomial is not irreducible. With our formulation we obtain some efficiency improvement.

Future Work an Open Problems

This thesis yields some open questions for future work.
A natural question is how to extend our technical result about the concrete hardness of the n-linear problem related to a CDH oracle. It would be very interesting to prove a lower bound in generic groups for the hardness of the n-linear problem if an adversary can make at most $n-1$ CDH oracle calls. Also, we would like to relate the hardness of the linear problems to the discrete logarithm problem. It this case we should also formalize the behavior of an adversary that has access to a discrete logarithm oracle. This latter problem seems not easy at all, since an adversary can perform all sort of operations on the string relative to the logarithm of an element (for example switching bits). This questions also leads to another problem, namely to give a computational assumption that turns out to be generically weaker than the CDH assumption but that still allows for building public-key encryption.

The last part of this work suggests other interesting aspects to study. The new decisional problem leads to two questions. An interesting problem is deciding whether it is possible to generalize our process and give the definition of a family of n-vector problems that get harder as n increases. If this is the case, we are immediately pushed to think about the linear assumptions and whether our family of assumptions can be related with them.

CHAPTER 1

Preliminaries

1.1. Notation and Definitions

If S is a set, we write $x \leftarrow S$ meaning that x is an element sampled uniformly from S.
If F is an event in a probability space Ω, we denote by $\bar{\mathsf{F}}$ the event that F does not occur.
We will write $\{0,1\}^\ell$, for $\ell \in \mathbb{N}$, in place of:

$$\underbrace{\{0,1\} \times \cdots \times \{0,1\}}_{\ell \text{ times}}.$$

We we write also $\{0,1\}^*$ for the set of bit strings of arbitrary length. If $x \in \{0,1\}^*$, we write $|x|$ for the length of the string x. If $x \in \mathbb{Z}_q^m$ is a vector, we write $x[i]$ for the i-th component of x.
If $(R, +, \cdot)$ is a ring, we write R^* to denote the subgroup of invertible elements of $(R, \cdot)$.

An **algorithm** is a finite sequence of operations that can be performed by a computer. A **probabilistic** algorithm is an algorithm that has internal access to a source of randomness that yields unbiased uniform random bits. A **deterministic** algorithm is an algorithm that has no randomness involved.
We write $y \leftarrow \mathcal{A}(x)$ meaning that $\mathcal{A}$ is a probabilist algorithm that on input x returns y. We write $y = \mathcal{A}(x)$ meaning that $\mathcal{A}$ is a deterministic algorithm that on input x outputs y. Often our algorithms will take 1^λ as input, meaning that the input is a string of 1's of length $\lambda \in \mathbb{N}$. In this case λ is called security parameter. The **running time** $t(n)$ of an algorithm is the maximum number of steps performed by the algorithm on any input of size n. If $\mathcal{A}$ is a probabilistic algorithm, then $t(n)$ is defined as the maximum number of steps on the internal randomness on any input of size n.

Definition 1.1.1. An algorithm $\mathcal{A}$ is said to run in **polynomial-time** if there exists a polynomial $p(t) \in \mathbb{Z}[t]$ such that, for every input $x \in \{0,1\}^*$, the computation of $\mathcal{A}(x)$ terminates in at most $p(|x|)$ steps.

Sometimes we will use also the term efficient for a polynomial-time algorithm.

Definition 1.1.2. A function $f : \mathbb{N} \to \mathbb{R}$ is **negligible** if, for every polynomial p, there exists an integer $n_p \in \mathbb{N}$ such that $f(n) < \frac{1}{p(n)}$ for every $n > n_p$.

Informally, we say that finding the solution to a certain problem is computationally infeasible if, for every polynomial-time algorithm, there exists a negligible function that bounds the probability of success of that algorithm.
For our purpose, a hash function is a function $H : \{0,1\}^* \to \{0,1\}^\ell$, for some fixed $\ell \in \mathbb{N}$.

1.2. Cryptographic Primitives

1.2.1. Public-Key Encryption Schemes.

DEFINITION 1.2.1. A **public-key encryption scheme** Π is a tuple of probabilistic polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

1. Gen is a probabilistic key-generation algorithm that takes as input 1^λ, for a security parameter λ in $\mathbb{N}$, and returns a public key pk and a secret key sk. The public key pk defines a space $\mathcal{M}$, called message space.
2. Enc is a probabilistic algorithm that takes as input a public key pk and a message m in $\mathcal{M}$ and returns a ciphertext c.
3. Dec is a deterministic algorithm that takes as input a secret key sk and a ciphertext c and returns a message m or the reject symbol $\perp$.

Moreover, a further fundamental property is required: correctness. We want that, for every λ in $\mathbb{N}$, every pair $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and for every message m in $\mathcal{M}$, the following equation holds:

$$\Pr[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m] = 1.$$

Usually we will write $\mathsf{Enc}_{pk}(m)$ in place of $\mathsf{Enc}(pk, m)$ and $\mathsf{Dec}_{sk}(c)$ in place of $\mathsf{Dec}(sk, c)$.

We now give the definitions of security for a public-key encryption scheme. First, we define security against chosen-plaintext attack [GM84]. Informally, a CPA-secure scheme is a scheme where the ciphertext does not leak any information about the plaintext. We formalize this security model by defining a game in which an adversary chooses two messages and a challenger encrypts one of them; the adversary wins the game if he can recognize which one of the two messages was encrypted.
To an algorithm $\mathcal{A}$ and a public-key encryption scheme Π we associate the following experiment $\mathsf{Exp}^{\mathrm{CPA}}_{\mathcal{A},\Pi}$ played between $\mathcal{A}$ and another algorithm called the challenger; the experiment is parametrized by a security parameter λ in $\mathbb{N}$:

1. The challenger generates a pair of keys (pk, sk) using the key-generation algorithm of Π and returns the public key pk to $\mathcal{A}$.
2. Adversary $\mathcal{A}$ outputs two messages m_0, m_1. The challenger chooses a uniform bit b and gives a challenger ciphertext $c^* \leftarrow \mathsf{Enc}_{pk}(m_b)$ back to $\mathcal{A}$.
3. Adversary $\mathcal{A}$ outputs a bit b'.

The output of this experiment is 1 if and only if $b' = b$.
We define the advantage of $\mathcal{A}$ as

$$\mathsf{AdvPub}^{\mathrm{CPA}}_{\mathcal{A},\Pi}(\lambda) = \left| \Pr[\mathsf{Exp}^{\mathrm{CPA}}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2} \right|.$$

Intuitively, The advantage of $\mathcal{A}$ measures the difference between the $\mathcal{A}$ wins and the probability that one wins just guessing naively.
Usually, in security experiments, we will denote by $*$ the challenge ciphertext.

DEFINITION 1.2.2. A public-key encryption scheme Π is (t, ε)**-secure under chosen plaintext attack** if, for every adversary $\mathcal{A}$ whose running time is bounded by t, the advantage $\mathsf{AdvPub}^{\mathrm{CPA}}_{\mathcal{A},\Pi}$ is bounded by the positive constant ε.

We now give another notion of security, namely security against chosen-ciphertext attack [NY90,RS92]. It will be clear soon that the latter definition leads to a stronger concept of security. In fact, the adversary against CPA-security is not interactive, while the adversary against CCA-security, that we are going to define below, has access to a decryption oracle used to decrypt any ciphertext.
To a public-key encryption scheme Π and an adversary $\mathcal{A}$, we associate the following experiment $\mathsf{Exp}^{\mathrm{CCA}}_{\mathcal{A},\Pi}$, played between $\mathcal{A}$ and a challenger:

1. The challenger generates a public and a secret key $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and gives the public key to $\mathcal{A}$.
2. $\mathcal{A}$ asks a number of decryption queries to the challenger: $\mathcal{A}$ sends a ciphertext c to the challenger which gives back the plaintext $m = \mathsf{Dec}_{sk}(c)$.
3. $\mathcal{A}$ sends two messages m_0, m_1 to the challenger. The challenger chooses a uniform bit b and sends the challenge ciphertext $c^* \leftarrow \mathsf{Enc}_{pk}(m_b)$ to $\mathcal{A}$.
4. $\mathcal{A}$ asks for more decryptions, as in step 2, but without the possibility of asking the decryption of the challenge ciphertext.
5. $\mathcal{A}$ outputs a bit b'.

This game outputs 1 if and only if $b' = b$.
We define the advantage of $\mathcal{A}$ as follows:

$$\mathsf{AdvPub}^{\mathrm{CCA}}_{\mathcal{A},\Pi}(\lambda) = \left| \Pr[\mathsf{PubK}^{\mathrm{CCA}}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2} \right|.$$

DEFINITION 1.2.3. A public-key encryption scheme Π is (Q_d, t, ε)**-secure against chosen ciphertext attack** if, for adversary $\mathcal{A}$ whose running time is bounded by t and that makes at most Q_d decryption queries, we the following inequality holds:

$$\mathsf{AdvPub}^{\mathrm{CCA}}_{\mathcal{A},\Pi}(\lambda) \leq \varepsilon.$$

1.2.2. Secret-Key Encryption Schemes. We now introduce the definition of symmetric encryption scheme. The main difference between public-key encryption and private-key encryption is that the latter one requires a shared secret key between the communicating parties. Usually, in our work, symmetric schemes will be used as internal components for public-key encryption schemes.

DEFINITION 1.2.4. A **private-key encryption scheme** (or symmetric scheme) Π is a tuple of probabilistic polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

1. Gen is a probabilistic algorithm, called key-generation algorithm, that takes as input 1^λ, for a security parameter $\lambda \in \mathbb{N}$, and returns a key k that specifies a message space $\mathcal{M}$.
2. Enc is a probabilistic algorithm that takes as input a key k and a message $m \in \mathcal{M}$ and returns a ciphertext c.
3. Dec is a deterministic algorithm that takes as input a key k and a ciphertext c and returns a message m or a reject symbol $\perp$.

We require that, for every $\lambda \in \mathbb{N}$, for every key $k \leftarrow \mathsf{Gen}(1^\lambda)$ and for every message m in the message space $\mathcal{M}$, we have $\Pr[\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m] = 1$.

As above, we will give two different notions of security for symmetric encryption schemes. The first one is analogous to the notion of CPA-security for public-key encryption schemes. To a secret-key encryption scheme Π and an algorithm $\mathcal{A}$ we associate the following experiment $\mathsf{Exp}^{\mathrm{OT}}_{\mathcal{A},\Pi}$ played between $\mathcal{A}$ and a challenger:

1. The challenger generates a key $k \leftarrow \mathsf{Gen}(1^\lambda)$ and sends the string 1^λ to $\mathcal{A}$.
2. Adversary $\mathcal{A}$ sends two messages m_0, m_1 of the same length to the challenger. The challenger chooses a uniform bit b and sends the challenge ciphertext $c^* \leftarrow \mathsf{Enc}_k(m_b)$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a bit b'.

The output of this experiment is 1 if and only if $b' = b$.
Once again we define the advantage of an algorithm $\mathcal{A}$ as:

$$\mathsf{AdvPriv}^{\mathrm{OT}}_{\mathcal{A},\Pi}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathrm{OT}}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2}\right|.$$

DEFINITION 1.2.5. A private-key encryption scheme Π is (t,ε)**-one-time secure** if, for every adversary $\mathcal{A}$ whose running time is bounded by t, the advantage $\mathsf{AdvPriv}^{\mathrm{OT}}_{\mathcal{A},\Pi}$ is bounded by the positive constant ε.

We now give the definition of one-time security against chosen-ciphertext attack. As happens in the case of public-key encryption schemes, this is a stronger notion of security than the previous one.
To an adversary $\mathcal{A}$ and a symmetric encryption scheme Π we associate the following experiment $\mathsf{Exp}^{\mathrm{CCA}}_{\mathcal{A},\Pi}$, played between $\mathcal{A}$ and another algorithm called the challenger:

1. The challenger generates a key $k \leftarrow \mathsf{Gen}(1^\lambda)$ and sends the string 1^λ to $\mathcal{A}$.
2. $\mathcal{A}$ sends two messages m_0, m_1 of the same length to the challenger. The challenger chooses a uniform bit b and sends the challenge ciphertext $c^* \leftarrow \mathsf{Enc}_k(m_b)$ to $\mathcal{A}$.
3. $\mathcal{A}$ asks for a number of decryption queries to the challenger: $\mathcal{A}$ sends a ciphertext c to the challenger and receives back a plaintext $m = \mathsf{Dec}_k(c)$ with the restriction $c \neq c^*$.
4. $\mathcal{A}$ outputs a bit b'.

This game outputs 1 if and only if $b' = b$.
We define the advantage of $\mathcal{A}$ as follows:

$$\mathsf{AdvPriv}^{\mathrm{CCA}}_{\mathcal{A},\Pi}(\lambda) = \left|\Pr[\mathsf{PubK}^{\mathrm{CCA}}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2}\right|.$$

DEFINITION 1.2.6. A private-key encryption scheme Π is (Q_d,t,ε)**-One-time secure against chosen-ciphertext attacks** (or OT-CCA secure) if, for every adversary $\mathcal{A}$ whose running time is bounded by t and that makes at most Q_d decryption queries, the advantage $\mathsf{AdvPriv}^{\mathrm{CCA}}_{\mathcal{A},\Pi}$ is bounded by the positive constant ε.

1.2.3. Message Authentication Codes. Message authentication codes are used in cryptography to gain authentication. Encryption schemes do not fulfill this task, so another tool is needed to allow an honest party to detect if a message was manipulated by a malicious part. As for symmetric encryption schemes, also message authentication codes require the honest parties to share a secret key. Formally:

DEFINITION 1.2.7. A **message authentication code** (MAC) consists of a tuple of probabilist polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Ver})$ such that:

1. The key-generation algorithm Gen takes as input 1^λ and outputs a secret key k that specifies a message space $\mathcal{M}$.
2. The probabilistic tag-generation algorithm Mac takes as input a pair (k,m), for a key k and a message $m \in \mathcal{M}$, and outputs a tag t.
3. The verification algorithm Ver takes as input a key k, a message m and a tag t and outputs a bit b. If $b = 1$ we say that t is a valid tag for m. Otherwise, we say that t is invalid.

We require that for every λ in $\mathbb{N}$, every k generated by $\mathsf{Gen}(1^\lambda)$ and every message $m \in \mathcal{M}$:

$$\Pr[\mathsf{Ver}(k, m, \mathsf{Mac}(k,m)) = 1] = 1.$$

We now give the notion of security for MACs [BKR00]. As we have done so far, we will give the definition of an experiment the models the interaction between and adversary and a honest party. To an algorithm $\mathcal{A}$ and a message authentication code Π we associate the following experiment $\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}$:

1. The challenger generates a secret key k using $\mathsf{Gen}(1^\lambda)$.
2. $\mathcal{A}$ is given 1^λ. Adversary $\mathcal{A}$ can ask for the tag for any message m.
3. Finally, $\mathcal{A}$ outputs a pair $(\hat{m}, \hat{t})$.

The output of this experiment is 1 if and only if $\mathsf{Ver}(k, \hat{m}, \hat{t}) = 1$ and $\hat{m}$ was not already submitted to the challenger for a tag in step 2.

DEFINITION 1.2.8. A message authentication code Π is **existentially unforgeable under an adaptive chosen-message attack** (or secure) if, for every polynomial-time algorithm $\mathcal{A}$, there exists a negligible function ε such that:

$$\Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \varepsilon(\lambda).$$

1.2.4. A Symmetric Encryption Scheme Secure against Chosen-Ciphertext Attack. In [CS03] we can find a construction of a CCA-secure symmetric encryption scheme. The idea lying at the base of this work consists of putting together a message authentication code along with a OT-secure symmetric encryption scheme. Informally, the sender and the receiver share two secret keys, one for a OT-secure encryption scheme and the other one for a MAC. When the sender has to encrypt a message m, he first encrypts it using the symmetric scheme getting a ciphertext c; then, he obtains a tag t using the MAC on c. The final ciphertext C consists of the pair (c, t). When the receiver has to decrypt a ciphertext (c, t), he decrypts c only if t is a valid tag for c. This scheme is CCA-secure because the message authentication code makes the decryption oracle useless. In fact, if the MAC is secure, the probability that the adversary forges a consistent pair (c, t), meaning that t is a valid tag for c, is negligible unless (c, t) was already given by the encryption oracle (and in this latter case, the adversary does not learn any further information). A formal proof is given in [CS03].

1.2.5. Hybrid Encryption. Besides public-key encryption schemes have many advantages over symmetric schemes, they also have some drawbacks. The most important issue is efficiency (in particular is the size of the message is big). That's why hybrid encryption [CS03] was introduced: combining private and public key encryption is possible to improve considerably the efficiency of an asymmetric scheme without losing the advantages given by public encryption. The basic idea of hybrid encryption consists in using a public key encryption scheme only to encrypt the secret key for a symmetric scheme. Assume a sender wants to encrypt a message m, then the procedure is the following:

1. The sender chooses a secret key k and encrypts it using the public key pk of the asymmetric scheme. Notice that the receiver will retrieve the key k using the secret key corresponding to pk, while any eavesdropper will be unable to get any information about k, thanks to the security properties of the public-key encryption scheme.
2. The sender encrypts the message m using the symmetric encryption scheme along with the secret key k. The receiver will decrypt the ciphertext using k.

As we said before, the power of hybrid encryption lays in the fact that we achieve the efficiency of symmetric encryption together with the functionality of public-key encryption. In fact, hybrid schemes are indeed public-key encryption scheme, since there is no need to share a secret key in advance.

The schemes that we will build in the next chapters are hybrid schemes and so they will internally use a symmetric scheme that we will call (E, D). We will not say much about these symmetric schemes, since one-time pads are sufficient for OT-secure symmetric schemes and since we already know a construction for a CCA-secure symmetric scheme (1.2.4). Moreover, we will use a special kind of hybrid encryption, called key encapsulation mechanism [Sho01], in which the key for the symmetric scheme is not chosen by the sender but it is computed during the encryption of the message.

1.2.6. Group-Generation Algorithm. In the construction of the key generation algorithm for a scheme, we will often use an algorithm that outputs a so-called group description. In our case, a group description corresponds to a 3-tuple $\mathscr{G} = (\mathbb{G}, q, g)$ where $\mathbb{G}$ is a cyclic group of prime order $q \geq 2^\lambda$ generated by g.

Definition 1.2.9. A **group generation algorithm** Ggen is a probabilistic polynomial-time algorithm that takes as input 1^λ, for some security parameter λ in $\mathbb{N}$, and outputs a group description $(\mathbb{G}, q, g)$, where $\mathbb{G}$ identifies a cyclic groups of prime order, q is the order of $\mathbb{G}$ and g is a generator.

A group-generation algorithm is always supposed to have efficient algorithms to test membership relations and to compute elementary group operations.
Before giving some examples of group descriptions, we explain the reason we prefer prime-order groups. In this thesis we will work with problems related to discrete logarithm, and even though computing discrete logarithm is believed to be hard also in non-prime order cyclic groups, it turns out that in these groups the problem can be reduced to a problem in a smaller group. In fact, some algorithms for discrete logarithm, as Pohlig-Hellman algorithm [PH78], reduce an instance in a group of order $q_1 \cdot q_2$ to two instances in groups of order q_1 and q_2 respectively. Another reason for working in prime order groups could be that in such groups every element, except the identity, is a generator. This fact can lead to faster algorithms that compute group generators.

The simplest example of a group description $(\mathbb{G}, q, g)$ is probably $((\mathbb{Z}_q, +), q, 1)$. Anyway, the additive group $\mathbb{Z}_q$ won't fit in our goal, since finding the discrete logarithm in groups of this form is trivial. In fact, if we are given $x \in Z_q$, the discrete logarithm is simply x. On the other hand, prime order subgroups of $(\mathbb{Z}_q^*, \cdot)$ are heavily used in cryptography. Another interesting example of groups in which the discrete logarithm is thought to be hard is given by elliptic curves. In fact, once we take an elliptic curve of the form $E : y = x^3 + ax + b$, for a, b in $\mathbb{Z}_q$ (where q a prime number), we can consider the set $E(\mathbb{Z}_q)$ of pairs (x, y) in $\mathbb{Z}_q^2$ that satisfy the equation E. It turns out that $E(\mathbb{Z}_q)$ is an abelian group having prime order with respect to the choice of a, b and q. For particular classes of elliptic curves, no non-generic algorithms are known to compute discrete logarithm. This means that, up to now, there are no sub-exponential algorithm for discrete logarithm in these curves. Another amazing property of elliptic curves is the fact that we can use smaller keys if we work with the group $E(\mathbb{Z}_q)$. For instance: for a 1024-bit prime q the time needed to compute the discrete logarithm in $\mathbb{Z}_q^*$, where sub-exponential time algorithms are knows, is the same time needed to solve the discrete logarithm in an elliptic curve group of order q', where q' is a prime of 132-bits [KL08].

1.3. Security in the Random Oracle Model

In the first part of this thesis we will work in the random oracle model [BR93]. This model is a useful tool to prove the security of cryptographic schemes. In the random oracle model we provide both the challenger and the adversary with a truly random function, thought

as a black box that takes as input a binary string x and returns a binary string y. The internal working of this box is unknown and everyone has the possibility to interact with it. When we are studying the security of a scheme in the random oracle model, we modify the experiments given above by letting the adversary has access to an oracle for the random function (i.e., the random oracle). In the security experiment the oracle can be simulated by maintaining a list of the previously asked values of the function. When a new value is asked, the game chooses a uniform element and stores it in the list for future use. Notice that this list will never be empty, since it must contain at least the value of the random function used for computing the challenge ciphertext.
In the real world, no random function exists; hence, when we want to build our schemes, an efficiently computable function will be used in its place. The hope is that these functions are good enough to well simulate a random function, so that we do not lose too much security. When an algorithm is working in the random oracle model we will specify the number of decryption queries, together with running time and success probability. For example, if Π is an encryption scheme, we say that Π is $(Q_d, Q_H, t, \varepsilon)$-CCA secure in the random oracle model meaning that for every algorithm $\mathcal{A}$ that runs in time at most t, that makes at most Q_d decryption queries and Q_H random oracle queries, the advantage $\mathsf{Adv}^{\mathrm{CCA}}_{\mathcal{A},\Pi}$ is less or equal to ε.

1.4. Proofs by Reduction

As we have seen in the definitions of security, in order to prove that a scheme is secure, we should show that it cannot be broken by any polynomial-time algorithm. However, such a proof would imply results in complexity that are far from being proved (for example that $\mathcal{NP} \neq \mathcal{NP}$). The common strategy to overcome this issue is assuming the "hardness" of some problems and then prove that breaking the scheme implies solving these problem. This kind of proof is called proof by reduction, as it reduces the issue of proving the security of a scheme to another problem. A proof by reduction works as follows: we assume that a problem X is computationally infeasible. Given this hypothesis, we want to prove that Π is a secure scheme. We fix a polynomial-time adversary $\mathcal{A}$ against the scheme Π. Then, we build another polynomial-time algorithm $\mathcal{B}$ trying to solve problem X using $\mathcal{A}$ as a subroutine. Algorithm $\mathcal{B}$ simulates the security-game for $\mathcal{A}$ and the algorithm $\mathcal{A}$ should not be able to recognize that it is not interacting with the scheme Π. The key point is that as long as $\mathcal{A}$ succeeds to break the scheme Π, the algorithm $\mathcal{B}$ should be able to solve the instance it was given. In this case, if the success probability of $\mathcal{A}$ in breaking Π is not negligible, so is the success probability of $\mathcal{B}$ in solving problem X. Since we are assuming that X is computationally infeasible, this fact leads to a contradiction. At this point, we say that the security of Π is based on the X assumption.

1.4.1. Tight Reductions vs. Loose Reductions. One of the aims of this work is presenting tight reductions. Informally, let Π be a scheme that bases its security on the problem X. Let $\mathcal{A}$ be an algorithm running in time $t_{\mathcal{A}}$ breaking scheme Π with probability $\varepsilon_{\mathcal{A}}$. If $\mathcal{A}$ can be used to build an algorithm $\mathcal{B}$ that has probability $\varepsilon_{\mathcal{B}} \approx \varepsilon_{\mathcal{A}}$ to solve problem X running in time $t_{\mathcal{B}} \approx t_{\mathcal{A}}$, we say that we have tight reduction. In our work we will always specify the relations between the success probabilities, giving explicit bounds, but we will not go into details about running times; we will simply say that the running times of our algorithms are approximatively the same, meaning that they differ by a polynomial in the number of elementary operations. Tight reductions are not only a theoretical issue in cryptography but they also have an important practical consequence: loose reductions mean loss of security and so larger keys to gain a satisfactory level of security (that can

be achieved with sensibly smaller keys if the reduction is tight). It is often the case that, due to a loose reduction, a scheme requires keys that are totally impractical. For example, consider that an exponentiation in a group where the elements are represented by k bit strings requires $\mathcal{O}(r^3)$ time. Hence, an exponentiation in 160-bit group is 4 times faster than an exponentiation in 256-bit group [BR09]. It is clear, then, that key-size is rather influent in the efficiency of a scheme.

1.5. Game-based Proofs

In this thesis, we will use the games technique introduced in [Sho04] to prove the security of our schemes. Game-based proofs use a sequence of games to pass from a game that would be hard to follow to another one that is trivial to analyze, but that is also really close to the initial game. While we give this sequence of games we keep the adversary fixed. We will usually start with the security game of a scheme and then we will give a number of games G_i with the property that, for every i, the games G_i and G_{i-1} differ in a negligible quantity. More precisely, at every step we will show that the difference between the probability that the adversary wins G_i and the probability that it wins G_{i-1} is negligible. We link each game with the previous one in this way, until we reach a game that is trivial to analyze. Gathering all the information of the intermediate steps, we can conclude that no adversary has a consistent advantage in breaking the scheme we are studying.
This games technique implies simple proofs, that are usually easy to follow. This is due to the fact that we can analyze the games proceeding by little steps.

The following Lemma 1.5.1 [Sho04] is an important tool to analyze the difference between two games. In fact, if two games proceed in the same way until an event (usually called "fail" event) happens, we can bound the difference between two games using the probability that this event happens.

THEOREM 1.5.1 (**Difference Lemma**). *Let* A, B *and* F *be three events defined in some probability space* Ω*. Suppose that "* $\mathsf{A} \wedge \bar{\mathsf{F}}$ *" holds if and only if "* $\mathsf{B} \wedge \bar{\mathsf{F}}$ *" holds, then:*

$$|\Pr[\mathsf{A}] - \Pr[\mathsf{B}]| \leq \Pr[\mathsf{F}].$$

PROOF. With a simple computation we get that:

$$\begin{aligned}
|\Pr[\mathsf{A}] - \Pr[\mathsf{B}]| &= |\Pr[\mathsf{A} \wedge \bar{\mathsf{F}}] + \Pr[\mathsf{A} \wedge \mathsf{F}] - \Pr[\mathsf{B} \wedge \bar{\mathsf{F}}] - \Pr[\mathsf{B} \wedge \mathsf{F}]| \\
&= |\Pr[\mathsf{A} \wedge \mathsf{F}] - \Pr[\mathsf{B} \wedge \mathsf{F}]| \\
&= |\Pr[\mathsf{A} \mid \mathsf{F}] \Pr[\mathsf{F}] - \Pr[\mathsf{B} \mid \mathsf{F}] \Pr[\mathsf{F}]| \\
&= |\Pr[\mathsf{F}] (\Pr[\mathsf{A} \mid \mathsf{F}] - \Pr[\mathsf{B} \mid \mathsf{F}])| \\
&\leq \Pr[\mathsf{F}]
\end{aligned}$$

The second equality holds because we are assuming that the probability that $\mathsf{A} \wedge \bar{\mathsf{F}}$ happens is the same that the event $\mathsf{B} \wedge \bar{\mathsf{F}}$ happens. Finally, we just notice that $\Pr[\mathsf{A}|\mathsf{F}] - \Pr[\mathsf{B}|\mathsf{F}] \leq 1$. □

1.6. Generic Group Model

The Generic Group given by Shoup in [Sho97] is a model where the adversary does not use any specific property of a given group $\mathbb{G}$. In order to achieve this property, we use a random embedding $\sigma : \mathbb{G} \to \{0,1\}^m$, for a sufficiently large m, and we give the adversary the encoding of the elements instead of the elements themselves. In this way, the only thing that the adversary can check is equality between strings. Since the adversary cannot compute group operations on its own, it is given an oracle that takes as input encoded elements and returns the encoded result (it can be the multiplication of two elements or the inverse of an

element). The oracle knows the random encoding, and as soon as it receives a string x it checks whether x is in the image $\sigma(\mathbb{G})$. If this is the case the oracle computes the pre-image $\sigma^{-1}(x)$ and use it to perform the required group operation; if x is not in the image the oracle returns a reject symbol ($\perp$). While working in generic groups, it is common to assume that the adversary asks group operations only on strings that he has previously received. We can do this without loss of generality because we are embedding our group in the set $\{0,1\}^m$, for a large m. This ensures that if the adversary chooses a uniform element y in $\{0,1\}^m$, the probability that y is in the image through σ is negligible. Hence, when the adversary gives this string y to the oracle, he will receive back $\perp$ with overwhelming probability.

When we will work in the generic group model, the adversary that simulates the group oracle will keep a list of order couples made of a polynomial and a bit string. At the beginning of the game the list will be populated by the pairs corresponding to the initial inputs of the problem we are studying. The polynomials in the list will be used to keep track of the bit strings while computing groups operations. Whenever the group oracle is called, new pairs will be added in the list.

One of the main uses of the generic group model is to study the hardness of a problem. This is exactly the reason we use this model in our work.
In the contest of Generic Groups, the following lemma [Sch80] will be very useful:

LEMMA 1.6.1. *Let q be a prime number. Let $F(X_1, \ldots, X_k)$ be a non-zero polynomial in $\mathbb{Z}_q[X_1, \ldots, X_k]$ of degree d. Then, if $x_1, \ldots, x_k$ are uniform elements of $\mathbb{Z}_q$, the probability that $F(x_1, \ldots, x_k) = 0$ is at most d/q*

1.6.1. Generic Bilinear Groups. Let $\mathbb{G}_0$ be a cyclic group of prime order q. A non-degenerate bilinear map is a map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ such that:

1. $e(x^a, y^b) = e(x, y)^{ab}$, for every x, y in $\mathbb{G}_0$ and for every a, b in $\mathbb{Z}_q$.
2. $e(x, y) = 1$ for every y in $\mathbb{G}_0$ implies $x = 1$.

If there exists a non-degenerate bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$, we refer to $\mathbb{G}_0$ as a bilinear group [BBG05]. It is sometimes useful to work in the so-called generic bilinear group model, in order to prove that a problem is hard to solve even if our group has a computable bilinear map. In fact, we know that the decisional Diffie-Hellman problem is hard in generic groups but this problem is easy as soon as we are working in a bilinear group.
The generic bilinear group model is essentially the same of the generic group model in which we give the adversary an oracle for a bilinear map. In this case, there will be two list of pairs, one for the elements of $\mathbb{G}_0$ and the other for elements of $\mathbb{G}_1$. We will have also a group oracle for operations in $\mathbb{G}_1$. Groups oracle queries for both groups are handle as in the generic group model.

CHAPTER 2

The Linear Diffie-Hellman Problems

In this chapter we give the definition of a family of problems called n-linear Diffie-Hellman Problems. They are a generalization of the Diffie-Hellman problem and we want to give a separation result between the CDH problem and the n-linear problem.

2.1. The Computational Diffie-Hellman Problem

Before introducing the n-linear problems, we briefly recall the Diffie-Hellman problem [DH76].
Let $\mathscr{G} = (\mathbb{G}, q, g)$ be a group description output by a group-generation algorithm Ggen. Given g, g^x and g^y, the Computational Diffie-Hellman (CDH) problem consists in computing g^{xy}.

To an algorithm $\mathcal{A}$ and a group-generation algorithm Ggen, we associate the following experiment $\mathsf{Exp}^{\mathrm{CDH}}_{\mathcal{A},\mathsf{Ggen}}$:

1. The challenger chooses a group description $\mathscr{G} = (\mathbb{G}, q, g) \leftarrow \mathsf{Ggen}(1^\lambda)$ and two uniform elements x, y in $\mathbb{Z}_q$.
2. $\mathcal{A}$ is given $\mathscr{G}$ and g^x, g^y. Finally, the algorithm $\mathcal{A}$ outputs an element h.

The experiment outputs 1 if and only if $h = g^{xy}$. We define the advantage of $\mathcal{A}$ in solving the CDH problem as: $\mathsf{Adv}^{\mathrm{CDH}}_{\mathcal{A},\mathsf{Ggen}}(\lambda) = \Pr[\mathsf{Exp}^{\mathrm{CDH}}_{\mathcal{A},\mathsf{Ggen}}(\lambda) = 1]$.

Definition 2.1.1. The computational Diffie-Hellman problem is (t, ε)-hard relative to Ggen if, for every algorithm $\mathcal{A}$ whose running time is bounded by t, the advantage $\mathsf{Adv}^{\mathrm{CDH}}_{\mathcal{A},\mathsf{Ggen}}$ is bounded by the positive constant ε.

2.2. The Linear Problems

Let $\mathscr{G} = (\mathbb{G}, q, g)$ be a group description output by a group-generation algorithm Ggen. Given $\mathscr{G}$ and uniform elements $g_1, \ldots, g_n, g_1^{x_1}, \ldots, g_n^{x_n}$ in $\mathbb{G}$, computing $g^{x_1+\cdots+x_n}$ is the computational n-linear Diffie-Hellman problem.

To an algorithm $\mathcal{A}$ and to a group-generation algorithm Ggen, we associate the following experiment $\mathsf{Exp}^{n\text{-}lin}_{\mathcal{A},\mathsf{Ggen}}$:

1. The challenger chooses a group description $(\mathbb{G}, q, g) \leftarrow \mathsf{Ggen}(1^\lambda)$ and n uniform elements $g_1, \ldots, g_n$ in $\mathbb{G}$. The challenger chooses $r_1, \ldots, r_n \leftarrow \mathbb{Z}_q$, too.
2. $\mathcal{A}$ is given $\mathscr{G}, g_1, \ldots, g_n, g_1^{r_1}, \ldots, g_n^{r_n}$ and outputs $h \in \mathbb{G}$.

The output of the experiment is 1 if and only if $h = g^{r_1+\cdots+r_n}$ and 0 otherwise. We define the advantage of adversary $\mathcal{A}$ as:

$$\mathsf{Adv}^{n\text{-}lin}_{\mathcal{A},\mathsf{Ggen}}(\lambda) = \Pr[\mathsf{Exp}^{n\text{-}lin}_{\mathcal{A},\mathsf{Ggen}}(\lambda) = 1].$$

Definition 2.2.1. The computational n-linear Diffie-Hellman problem is (t, ε)-hard relative to Ggen if, for every algorithm $\mathcal{A}$ whose running time is bounded by t, the advantage $\mathsf{Adv}^{n\text{-}lin}_{\mathcal{A},\mathsf{Ggen}}$ is bounded by the positive constant ε.

As we have noticed above, the n-linear problems are a generalization of the CDH problem. In fact, it is easy to see that the 1-linear problem is exactly the CDH problem.

LEMMA 2.2.2. *The 1-linear problem is tightly equivalent to the CDH problem. More precisely, the 1-linear problem is (t, ε)-hard relative to* Gen *if and only if the CDH problem is (t, ε)-hard relative to* Gen.

PROOF. ($\Rightarrow$) Assume that $\mathcal{A}$ is an algorithm that solves the CDH problem with success probability ε. We can build an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine against the 1-linear problem as follows: when $\mathcal{B}$ receives $\mathscr{G}, g^x, g^{xy}$ it invokes adversary $\mathcal{A}$ on input $\mathscr{G}' = (\mathbb{G}, q, h), h^{(x^{-1})}, h^y$, where $h = g^x$. Then, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. Since $h^{x^{-1}y} = g^y$, every time $\mathcal{A}$ solves correctly its instance of the CDH problem, the algorithm $\mathcal{B}$ outputs a correct solution to the 1-linear problem. This means that the success probability of $\mathcal{B}$ is at least ε. Moreover we notice that the running time of the two algorithms is the same.
($\Leftarrow$) Conversely, let $\mathcal{B}$ be an algorithm solving the 1-linear problem with probability ε. We construct an algorithm $\mathcal{A}$ against the CDH problem as follows: when $\mathcal{A}$ receives $\mathscr{G}, g^x, g^y$ as input, it invokes $\mathcal{B}$ on input $\mathscr{G}' = (\mathbb{G}, q, h), h^{(x^{-1})}, h^{(x^{-1})y}$, where $h = g^x$. The algorithm $\mathcal{A}$ outputs whatever $\mathcal{B}$ outputs. Since $h^y = g^{xy}$ we have that $\mathcal{A}$ solves correctly the CDH problem with probability at least ε. Again, we have that the running times of $\mathcal{A}$ and $\mathcal{B}$ are equal. This proves tight equivalence. □

2.3. The Strong Linear Problems

We now introduce another linear problem, the strong linear problem. We will use it to establish the CCA-security of our scheme, since, as we will see later, the n-linear problem is not sufficient.
We start giving the definition of the predicate called *n-lin*.

DEFINITION 2.3.1. Let $\mathscr{G} = (\mathbb{G}, q, g)$ be a group description output by a group-generation algorithm Gen. Let $g_1, \ldots, g_n, g_1^{x_1}, \ldots, g_n^{x_n}$ be elements of $\mathbb{G}$. The predicate

$$n\text{-}lin_{g,g_1,\ldots,g_n}(g_1^{x_1}, \ldots, g_n^{x_n}, Z)$$

returns 1 if and only if $Z = g^{x_1+\cdots+x_n}$ and 0 otherwise.

Let $\mathscr{G}$ be a description. Given $\mathscr{G}$ together with $g_1, \ldots, g_n, g_1^{x_1}, \ldots, g_n^{x_n}$ and an oracle for the predicate $n\text{-}lin_{g,g_1,\ldots,g_n}$, finding $g^{x_1+\cdots,+x_n}$ is the strong n-linear problem. To an adversary $\mathcal{A}$ and a group-generation algorithm Gen, we associate the following experiment $\mathsf{Exp}^{Sn\text{-}lin}_{\mathcal{A},\mathsf{Gen}}$:

1. The challenger chooses a group description $\mathscr{G} = (\mathbb{G}, q, g) \leftarrow \mathsf{Gen}(1^\lambda)$ and n uniform elements $g_1, \ldots, g_n$ in $\mathbb{G}$. The challenger also chooses $r_1, \ldots, r_n \leftarrow \mathbb{Z}_q$.
2. $\mathcal{A}$ is given $\mathscr{G}, g_1, \ldots, g_n, g_1^{r_1}, \ldots, g_n^{r_n}$.
3. $\mathcal{A}$ has access to an oracle for the predicate $n\text{-}lin_{g,g_1,\ldots,g_n}$.
4. $\mathcal{A}$ outputs $h \in \mathbb{G}$.

The output of the experiment is 1 if and only if $h = g^{r_1+\cdots+r_n}$ and 0 otherwise. We define the advantage $\mathsf{Adv}^{Sn\text{-}lin}_{\mathcal{A},\mathsf{Gen}}(\lambda)$ of $\mathcal{A}$ to be equal to $\Pr[\mathsf{Exp}^{Sn\text{-}lin}_{\mathcal{A},\mathsf{Gen}}(\lambda) = 1]$.

DEFINITION 2.3.2. The strong n-linear Diffie-Hellman problem is (Q, t, ε)-hard relative to Gen if, for every algorithm $\mathcal{A}$ whose running time is bounded by t and that makes at most Q queries to the oracle for the predicate $n\text{-}lin$, the advantage $\mathsf{Adv}^{Sn\text{-}lin}_{\mathcal{A},\mathsf{Gen}}$ is bounded by the positive constant ε.

2.4. Complexity

As for many computational problems related to the DH problem, even the n-linear problems are equivalent to the CDH problem (in the asymptotic sense). However, it will be clear that we cannot tightly bound the advantage of the adversary against the n-linear problem using an adversary against the CDH problem. We stress the fact that n must be a polynomial in the security parameter λ. If n was exponential in λ, the problem would be intractable for polynomial-time every adversary, but it would not be useful for any cryptographic purpose. Since we will problem the following theorem in the asymptotic setting, we give here the definition of n-linear assumption.

DEFINITION 2.4.1. The n-**linear assumption** relative to Ggen holds if, for every polynomial time algorithm $\mathcal{A}$, there exists a negligible function ε such that $\mathsf{Adv}^{n\text{-}lin}_{\mathcal{A},\mathsf{Ggen}}(\lambda) \leq \varepsilon(\lambda)$.

LEMMA 2.4.2. *The n-linear assumption is equivalent to the CDH assumption.*

PROOF.
CDH implies n-linear. Suppose that the CDH assumption holds relative to Ggen. Let $\mathcal{A}$ be an adversary against the n-linear problem that succeeds with non-negligible probability. We construct an adversary $\mathcal{B}$ against the CDH assumption that uses $\mathcal{A}$ as a subroutine as follows: when $\mathcal{B}$ receives its input $\mathscr{G}, g^x, g^y$, it chooses uniform elements $s_2, \ldots, s_n, t_2, \ldots, t_n$ in $\mathbb{Z}_q$ and computes $g_i = g^{s_i}$ for i in $\{2, \ldots, n\}$. Then, $\mathcal{B}$ invokes $\mathcal{A}$ on input $(\mathscr{G}' = (\mathbb{G}, q, g^x), g, g_2, \ldots, g_n, g^y, g_2^{t_1}, \ldots, g_n^{t_n})$. When $\mathcal{A}$ outputs h', adversary $\mathcal{B}$ computes $h = h' \prod_{i=2}^{n} (g^{xt_i})^{-1}$. Finally $\mathcal{B}$ outputs h. If $\mathcal{A}$ gives the correct answer to the n-linear problem, we have that:

$$h = h' \prod_{i=2}^{n} (g^{xt_i})^{-1} = (g^x)^{y+t_2\cdots+t_n} \prod_{i=2}^{n} (g^{xt_i})^{-1} = g^{xy}.$$

This means that every time $\mathcal{A}$ solves correctly its instance, adversary $\mathcal{B}$ outputs a correct solution. Hence: the success probability of $\mathcal{B}$ is greater or equal to the success probability of $\mathcal{A}$. This is a contradiction, since we are assuming that the CDH assumption holds. Hence, the CDH assumption implies the n-linear assumption.
n-linear implies CDH. To prove the converse we proceed in the same way. Assume the n-linear assumption holds relative to Ggen and let $\mathcal{B}$ be an adversary solving the CDH problem with non-negligible. Using the following theorem [Sho97], we can assume that the success probability of $\mathcal{B}$ is negligibly close to 1:

THEOREM 2.4.3. *Given an algorithm against the Diffie-Hellman problem that has non-negligible success probability ε, we can construct a probabilistic algorithm for the CDH problem with the following property: for given α, with $0 < \alpha < 1$, for all inputs, the output of the algorithm is correct with probability at least $1 - \alpha$.*

Consider, now, the following adversary $\mathcal{A}$ against the n-linear assumption: when $\mathcal{A}$ receives its input $(\mathscr{G}, g_1, \ldots, g_n, g_1^{x_1}, \ldots, g_n^{x_n})$ it chooses n elements $s_1, \ldots, s_n$ uniformly in $\mathbb{Z}_q$. Then $\mathcal{A}$ invokes adversary $\mathcal{B}$ on input $((\mathbb{G}, q, g_i), g_i^{s_i}, g_i^{x_i})$ for $i \in \{0, 1\}$ n consecutive times. When $\mathcal{B}$ receives the outputs $h_1, \ldots, h_n$ from $\mathcal{A}$, it computes $h'_i = h_i^{(s_i)^{-1}}$. Finally $\mathcal{B}$ outputs $h' = h'_1 \cdots h'_n$.
We notice that, if $\mathcal{A}$ responds correctly to all the n invocations, then $\mathcal{B}$ solves correctly its instance of the n-linear problem. This means that:

$$\mathsf{Adv}^{n\text{-}lin}_{\mathcal{A},\mathsf{Ggen}}(\lambda) \geq \left(\mathsf{Adv}^{\mathrm{CDH}}_{\mathcal{B},\mathsf{Ggen}}(\lambda)\right)^n. \tag{2.1}$$

At this point, we recall that we can assume $\mathsf{Adv}^{\mathrm{CDH}}_{\mathcal{B},\mathsf{Ggen}} = 1 - \alpha$, for a negligible function α. Hence, also $\left(\mathsf{Adv}^{\mathrm{CDH}}_{\mathcal{B}}(\lambda)\right)^n$ would be non negligible, for any n polynomial in λ. By equation (2.1), we will have that $\mathsf{Adv}^{n\text{-}lin}_{\mathcal{A}}(\lambda)$ is non negligible, which is a contradiction to our hypothesis. □

If $\mathscr{G} = (\mathbb{G}, q, g)$ is a group description, a **Diffie-Hellman oracle** is an algorithm that, on input g, g^x, g^y, returns g^{xy}. We could as well define a DH-oracle as an algorithm that, on input g, g^x, g^{xy} returns g^y. By Lemma 2.2.2 these two definitions are equivalent.
In Lemma 2.4.2 we proved that solving n instances of the CDH problem is at least as hard as solving the n-linear problem. We conjecture that there is an equivalence, namely: solving the n-linear problem is tightly equivalent to solving n instances of the CDH problem. To prove this equivalence we should prove that an adversary working in generic groups that can use at most $n-1$ calls to a CDH oracle is not able to solve the n-linear problem with not negligible probability. At the moment we can prove are conjecture just for $n = 2$.

THEOREM 2.4.4. *Let $\mathbb{G}$ be a cyclic group of prime order q and let σ be a random encoding for $\mathbb{G}_0$. For every adversary $\mathcal{A}$ that solves the 2-linear problem in generic groups making at most Q queries to the group operation oracle and at most 1 query to the CDH oracle, we have:*

$$\Pr\left[\mathcal{A}\left(\begin{array}{c} q, \sigma(1), \sigma(x_1), \sigma(x_2), \\ \sigma(x_1y_1), \sigma(x_2y_2) \end{array}\right) = \sigma(y_1+y_2) \,\middle|\, x_1, x_2, y_1, y_2 \leftarrow \mathbb{Z}_q\right] \leq \frac{2(Q^2+13Q+42)}{q}.$$

PROOF. Consider an algorithm $\mathcal{B}$ that plays the following game with $\mathcal{A}$. The algorithm $\mathcal{B}$ maintains a list $L = \{(F_i, \sigma_i)\}$, where σ_i are the encoded strings that $\mathcal{B}$ gives to $\mathcal{A}$ and, to keep track of these strings, $\mathcal{B}$ uses polynomials F_i in $\mathbb{Z}_q(X_1, X_2, Y_1, Y_2)$. Initially, the list L consists of

$$\{(1, \sigma(1)), (X_1, \sigma(x_1)), (X_2, \sigma(x_2)), (X_1Y_1, \sigma(x_1y_1)), (X_2Y_2, \sigma(x_2y_2))\}$$

which corresponds to the input of the 2-linear problem.
As we pointed out in Section 1.6, we can assume without loss of generality that algorithm $\mathcal{A}$ asks $\mathcal{B}$ to compute group operations only on strings that are already contained in L.

Group action. When $\mathcal{A}$ asks for multiplication of two elements σ_i and σ_j, algorithm $\mathcal{B}$ recovers the corresponding polynomial F_i and F_j and computes $F_i + F_j$. If the polynomial $F_i + F_j$ is already in the list, $\mathcal{B}$ returns the corresponding string to $\mathcal{A}$. Otherwise, $\mathcal{B}$ returns a uniform string σ' in $\{0,1\}^m$ and stores $(F_i + F_j, \sigma')$ in the list.

Inversion. When $\mathcal{A}$ asks for the inversion of an element σ_i the algorithm $\mathcal{B}$ recovers the corresponding polynomial F_i and computes $-F_i$. At this point, if $-F_i$ is already in the list, $\mathcal{B}$ return the corresponding string. Otherwise, $\mathcal{B}$ returns a uniform string and store it together with $-F_i$ in the list.

CDH Oracle. We recall briefly the behavior of CDH oracle. If g, g^x, g^{xy} are elements of a group $\mathbb{G}$, then the CDH oracle on input (g, g^x, g^{xy}) will output g^y. We can extend the CDH oracle to a generic base as follows: if the input is (g^a, g^b, g^c) the outputs will be $g^{ac/b}$. In fact, let $A = g^a, B = g^b$ and $C = g^y$, then $B = A^{b/a}$ and $C = B^{c/b}$; it is clear, then, the output is exactly $A^{c/b} = g^{ac/b}$.
When $\mathcal{A}$ invokes the CDH oracle on input $\sigma_i, \sigma_j, \sigma_k$, algorithm $\mathcal{B}$ recovers the corresponding polynomials F_i, F_j and F_k and computes $\frac{F_iF_k}{F_j}$. Next, $\mathcal{B}$ checks whether this polynomial is

in the list or not. In the first case, $\mathcal{B}$ returns the corresponding string. Otherwise, $\mathcal{B}$ returns a uniform string and stores it in the list together with $\frac{F_i F_k}{F_j}$.

After at most Q queries, adversary $\mathcal{A}$ will output some string σ. At this point, algorithm $\mathcal{B}$ chooses x_1, x_2, y_1, y_2 uniformly at random in $\mathbb{Z}_q$. We notice that the simulation provided by $\mathcal{B}$ is perfect and reveals no information about $\sigma(y_1+y_2)$ unless one of the two cases below happens:

1. $F_i(x_1, x_2, y_1, y_2) = F_j(x_1, x_2, y_1, y_2)$ but $F_i \neq F_j$
2. $F_i(x_1, x_2, y_1, y_2) = y_1 + y_2$

If (1) or (2) happens, we say that we have a collision.

Claim: algorithm $\mathcal{A}$ is not able to create a polynomial of the form $Y_1 + Y_2$.
For the moment assume that this claim holds; we will prove it later.
Now, we only have to bound the possibility that a random choice of elements x_1, x_2, y_1, y_2 causes a collision of the type (1) or (2). We notice that all the polynomials in the table have degree at most 4, so the probability that two different polynomials have the same value after the substitution, i.e $F_i(x_1, x_2, y_1, y_2) = F_j(x_1, x_2, y_1, y_2)$ but $F_i \neq F_j$, is at most $4/q$, by Lemma 1.6.1. For the same reason, the probability that the value of one polynomial F_i is equal to $y_1 + y_2$ is at most $4/q$. A sum over all pairs of entries of the list gives a bound for the probability that (1) or (2) happens. Since the list L contains $Q + 6$ elements, this probability is at most:

$$\begin{aligned}
&\binom{Q+6}{2}\frac{4}{q} + (Q+6)\frac{4}{q} \\
=&\left(\frac{(Q+6)(Q+5)}{2} + (Q+6)\right)\frac{4}{q} \\
=&\frac{2(Q^2+13Q+42)}{q}
\end{aligned}$$

This bound the probability of success of adversary $\mathcal{A}$.

Proof of the Claim.
To prove our claim , we notice that is sufficient to show that after the CDH oracle $\mathcal{A}$ is not able to obtain a polynomial of the form

$$\omega(Y_1 + Y_2) + \alpha + \beta X_1 + \gamma X_2 + \delta X_1 Y_1 + \varepsilon X_2 Y_2$$

where $\alpha, \beta, \gamma, \delta, \varepsilon$ and ω are in $\mathbb{F}_q$. Let F_i, F_j and F_k be three polynomials of the list. We know that the CDH oracle is computed by $F = \frac{F_i F_k}{F_j}$. Since the polynomials F_i, F_j and F_k are in $\mathbb{F}_q[X_1, X_2, Y_1, Y_2]$, we have that F is of the form:

$$F = \frac{F_i F_k}{F_j} = \frac{\overbrace{(\alpha_1 + \beta_1 x_1 + \gamma_1 x_2}^{R_1} + \overbrace{\delta_1 x_1 y_1 + \varepsilon_1 x_2 y_2)}^{R_2}\overbrace{(\alpha_2 + \beta_2 x_1 + \gamma_2 x_2}^{S_1} + \overbrace{\delta_2 x_1 y_1 + \varepsilon_2 x_2 y_2)}^{S_2}}{(\alpha_3 + \beta_3 x_1 + \gamma_3 x_2 + \delta_3 x_1 y_1 + \varepsilon_3 x_2 y_2)}$$

where all the $\alpha_i, \beta_i, \gamma_i, \delta_i, \varepsilon_i$ are element of $\mathbb{F}_q$.
We split the product in the numerator in four parts, and analyze them one by one.
In the term given by $R_1 S_1$ does not appear any Y's, so we can ignore it.
In $R_2 S_2$ we notice that both Y_1 and Y_2 are multiply by terms of degree 3. To obtain a polynomial of he form $\omega(Y_1 + Y_2) + \alpha + \beta X_1 + \gamma X_2 + \delta X_1 Y_1 + \varepsilon X_2 Y_2$, we have to cancel out the terms that multiply Y_1 and Y_2. The only way to do this is cancel these the terms

using the denominator. Anyway, we notice that the denominator has degree at most two, so it cannot cancel out terms of degree 3.
Finally, the term $R_1S_2 + R_2S_1$ is equal to:

$$(\alpha_1 + \beta_1 X_1 + \gamma_1 X_2)(\delta_2 X_1 Y_1 + \varepsilon_2 X_2 Y_2) + (\delta_1 X_1 Y_1 + \varepsilon_1 X_2 Y_2)(\alpha_2 + \beta_2 X_1 + \gamma_2 X_2) =$$
$$= Y_1[a_1 X_1 + b_1 X_1^2 + c_1 X_1 X_2] + Y_2[a_2 X_2 + b_2 X_2^2 + c_2 X_1 X_2]$$

for some coefficients $a_1, b_1, c_1, a_2, b_2, c_2$ in $\mathbb{F}_q$. To obtain $Y_1 + Y_2$ we have to gather the common factors that multiplies both Y_1 and Y_2. We notice that the only factor that appears in both parts in X_1X_2. But we can see also that in the denominator this factor does not appears, so we cannot cancel it out an obtain $Y_1 + Y_2$.
This shows that $\mathcal{A}$ is not able itself to find the solution to the 2-linear problem.

□

CHAPTER 3

A Scheme Based on the Strong Linear Problem

In this chapter we will present a scheme that bases its security on the problems described in Chapter 2. Next, we will study both CPA and CCA-security and we will show that to reach the latter one we have to use the strong linear problem.

3.1. The scheme

CONSTRUCTION 3.1.1. Let Ggen be a group generation algorithm. Let (E, D) be a symmetric encryption scheme with key space $\mathcal{K} = \{0,1\}^\ell$ and message space $\mathcal{M}$. Consider the following public-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$:

- The key-generation algorithm Gen, upon input 1^λ, generates a group description $\mathscr{G} = (\mathbb{G}, q, g) \leftarrow \mathsf{Ggen}(1^\lambda)$ and a hash function $H : \{0,1\}^* \to \{0,1\}^\ell$. Then it samples n uniform elements $x_1, \ldots, x_n$ in $\mathbb{Z}_q^*$ and outputs a secret key $sk = (x_1, \ldots, x_n)$ and a public key $pk = (\mathscr{G}, X_1, \ldots, X_n, H)$, where $X_i = g^{x_i}$ for every $i \in \{1, \ldots, n\}$.
- The encryption algorithm Enc, upon input a message $m \in \mathcal{M}$ and a public key pk, chooses $y_1, \ldots, y_n$ in $\mathbb{Z}_q$ uniformly and computes $Y_i = X_i^{y_i}$ for every i in $\{1, \ldots, n\}$. Next, it sets $Z = g^{y_1 + \cdots + y_n}$, $k = H(Y_1, \ldots, Y_n, Z)$ and $c \leftarrow \mathsf{E}_k(m)$. The ciphertext is $(Y_1, \ldots, Y_n, c)$.
- The decryption algorithm Dec, upon input a ciphertext $(Y_1, \ldots, Y_n, c)$ and a secret key $sk = (x_1, \ldots, x_n)$, computes

$$Z = Y_1^{(x_1^{-1})} \cdots Y_n^{(x_n^{-1})}, \quad k = H(Y_1, \ldots, Y_n, Z).$$

  The plaintext is $m = \mathsf{D}_k(c) \in \mathcal{M} \cup \{\perp\}$.

LEMMA 3.1.2. *The scheme* Π *is correct.*

PROOF. To prove correctness, we have to show that, for every

$$pk = (\mathscr{G}, X_1 = g^{x_1}, \ldots, X_n = g^{x_n}, H)$$

generated by Gen and for every $y_1, \ldots, y_n$ in $\mathbb{Z}_q$, the way we compute Z in encryption is equivalent to the way we compute Z in decryption. This means that we have to show that $Y_1^{(x_1^{-1})} \cdots Y_n^{(x_n^{-1})} = g^{y_1 + \cdots + y_n}$, where $Y_i = X_i^{y_i}$. We have that:

$$\begin{aligned} Y_1^{(x_1^{-1})} \cdots Y_n^{(x_n^{-1})} &= (X_1^{y_1})^{(x_1^{-1})} \cdots (X_n^{y_n})^{(x_n^{-1})} \\ &= g^{y_1 x_1 (x_1^{-1})} \cdots g^{y_n x_n (x_n^{-1})} \\ &= g^{y_1 + \cdots + y_n} \end{aligned}$$

□

3.2. Security against Chosen-Plaintext Attack

We prove now that scheme Π is CPA-secure assuming the hardness of the n-linear problem.

THEOREM 3.2.1. *If H is modeled as a random oracle, (E,D) is a (t',ε')-OT-secure symmetric encryption scheme and the n-linear problem is (t'',ε'')-hard relative to Ggen, then Π from construction 3.1.1 is (Q_H,t,ε)-secure under chosen-plaintext attack, where $t \approx t' \approx t''$ and $\varepsilon \leq \varepsilon' + Q_H \cdot \varepsilon''$.*

PROOF. We will prove the theorem using the games technique. We take an adversary $\mathcal{A}$ against scheme Π running in time t and we keep it fixed in all the following games.

Game 0. The first game is the standard game for CPA-security defined in 1.2.2. Let us recall it briefly. At the beginning of the game, after having computed the public and the secret key $(pk,sk) = \big((\mathscr{G},X_1\ldots,X_n,H),(x_1,\ldots,x_n)\big)$, the challenger chooses $y_1,\ldots,y_n$ uniformly in $\mathbb{Z}_q$ and computes $(Y_1^*,\ldots,Y_n^*) = (X_1^{y_1},\ldots,X_n^{y_n})$ and $Z^* = g^{y_1+\cdots+y_n}$. Then the challenger computes $k^* = H(Y_1^*,\ldots,Y_n^*,Z^*)$. When $\mathcal{A}$ outputs two messages m_0,m_1, the challenger gives $\mathcal{A}$ the challenge ciphertext $(Y_1^*,\ldots,Y_n^*,\mathsf{E}_{k^*}(m_b))$. Finally $\mathcal{A}$ outputs its guess b'. We recall that $\mathcal{A}$ has access to an oracle for the hash function H throughout the game.

Game 1. Game 1 proceeds as Game 0 until adversary $\mathcal{A}$ outputs two messages (m_0,m_1). At this point the challenger chooses a uniform k in $\{0,1\}^\ell$ and use it to compute the challenge ciphertext $(Y_1^*,\ldots,Y_n^*,\mathsf{E}_k(m_b))$. The rest of the game proceeds exactly like Game 0.

Let G_i be the event that $\mathcal{A}$ wins in Game i.

We notice that Game 0 and Game 1 proceed identically until $\mathcal{A}$ asks for the value of the hash function in $(Y_1^*,\ldots,Y_n^*,Z^*)$. If we call this event F, then by the Difference Lemma 1.5.1 we have:

$$|\Pr[G_0(\lambda)=1] - \Pr[G_1(\lambda)=1]| \leq \Pr[\mathsf{F}].$$

Claim: the probability that F happens is less or equal to $Q_H \cdot \varepsilon''$.

To prove the claim we will build an adversary $\mathcal{B}$ against the n-linear problem that runs in approximately the same time of $\mathcal{A}$ and that is successful when event F happens. Adversary $\mathcal{B}$ works as follows: upon input $(\mathscr{G},g_1,\ldots,g_n,g_1^{x_1},\ldots,g_n^{x_n})$, adversary $\mathcal{B}$ chooses a hash function H and creates a table whose entries will be of the type $(Y_1,\ldots,Y_n,Z,k)$, where the entry k is equal to the value of the hash function H in $(Y_1,\ldots,Y_n,Z)$. Next, $\mathcal{B}$ invokes $\mathcal{A}$ on input $pk = (\mathscr{G},g_1,\ldots,g_n,H)$. When $\mathcal{A}$ asks for the value of the hash function in $(Y_1,\ldots,Y_n,Z)$, adversary $\mathcal{B}$ answers as follows:

- If there is an entry $(Y_1,\ldots,Y_n,Z,k)$ in the table, then $\mathcal{B}$ returns k.
- Otherwise, $\mathcal{B}$ picks a uniform element k in $\{0,1\}^\ell$, returns k to $\mathcal{A}$ and stores $(Y_1,\ldots,Y_n,Z,k)$ in the table.

When $\mathcal{A}$ outputs two messages m_0,m_1, adversary $\mathcal{B}$ chooses a bit b and a uniform k in $\{0,1\}^\ell$ and returns the challenge ciphertext $(g_1^{x_1},\ldots,g_n^{x_n},\mathsf{E}_k(m_b))$. Finally, when $\mathcal{A}$ outputs its guess b', adversary $\mathcal{B}$ chooses one entry $(Y_1,\ldots,Y_n,Z)$ uniformly in the list and outputs Z.

If event F happens, the list will contain an entry of the form $(Y_1^*,\ldots,Y_n^*,Z^*)$, where Z^* is the correct solution to the n-linear problem. This means that, in this case, adversary $\mathcal{B}$ will output a correct solution with probability at least $1/Q_H$, where Q_H is the number of queries performed by $\mathcal{A}$ to the random oracle. This means that: $\mathsf{Adv}^{n\text{-}lin}_{\mathcal{B},\mathsf{Ggen}} \geq \Pr[\mathsf{F}]/Q_H$. Since, by hypothesis, the n-linear problem is (t'',ε'')-hard and the running times of $\mathcal{A}$ and

$\mathcal{B}$ are approximately the same, we have that:

$$\Pr[\mathsf{F}] \leq Q \cdot \varepsilon''. \tag{3.1}$$

To conclude the proof we state the following claim:
Claim: The advantage of $\mathcal{A}$ in Game 1 is bounded by ε'.
To prove this fact, we will build an adversary $\mathcal{A}'$ against the symmetric scheme (E, D) that uses $\mathcal{A}$ as a subroutine. Adversary $\mathcal{A}'$ is constructed as follows: upon input 1^λ, it generates a pair $(pk, sk) = \big((\mathscr{G}, X_1 \ldots, X_n, H), (x_1, \ldots, x_n)\big)$ using the key-generation algorithm of Π. Then, it chooses n uniform elements $y_1, \ldots, y_n$ in $\mathbb{Z}_q$ and sets $(Y_1^*, \ldots, Y_n^*, Z^*) = (X_1^{y_1}, \ldots, X_n^{y_n}, g^{y_1 + \cdots + y_n})$. Adversary $\mathcal{A}'$ simulated the oracle for the hash function in the usual way. When $\mathcal{A}$ outputs two messages m_0, m_1, adversary $\mathcal{A}'$ gives them to the challenger. If c^* is the challenge ciphertext for $\mathcal{A}'$, then $\mathcal{A}'$ gives $(Y_1^*, \ldots, Y_n^*, c^*)$ as ciphertext to $\mathcal{A}$. Finally $\mathcal{A}'$ outputs whatever $\mathcal{A}$ outputs.
We notice that $\mathcal{A}'$ provides a perfect simulation of Game 1 to $\mathcal{A}$. Also, if $\mathcal{A}$ wins Game 1, adversary $\mathcal{A}'$ outputs a correct guess of the bit b. Hence:

$$\mathsf{Adv}^{\mathrm{OT}}_{\mathcal{A}', (\mathsf{E},\mathsf{D})}(\lambda) \geq \left| \Pr[G_1(\lambda) = 1] - \frac{1}{2} \right|.$$

Finally, we notice that the running time of $\mathcal{A}'$ is approximately the same of the running time of $\mathcal{A}$. Since, by hypothesis, the scheme (E, D) is (t', ε')-OT-secure, we can conclude that:

$$\left| \Pr[G_1(\lambda) = 1] - \frac{1}{2} \right| \leq \varepsilon' \tag{3.2}$$

Putting together equations (3.1) and (3.2) we have that Π is a (t, ε)-CPA secure scheme, where:

$$\varepsilon \leq Q_H \cdot \varepsilon'' + \varepsilon'.$$

□

3.3. Security against Chosen-Ciphertext Attack

In Section 3.2 we proved that the hardness of the n-linear problem is sufficient to gain security against chosen-plaintext attack. In this section we will see that this assumption does not grant security against chosen-ciphertext attack. Actually, we will prove something more general, namely that breaking scheme Π, under some assumptions, is equivalent to solving the strong n-linear problem.

THEOREM 3.3.1. *Let H be modeled as a random oracle and let (E, D) be a CCA-secure symmetric encryption scheme. Then, breaking scheme Π from construction 3.1.1 is tightly equivalent to solving the strong n-linear problem.*

LEMMA 3.3.2. *Let (E, D) be a (t, ε)-OT-CCA secure encryption scheme (where t is bigger then the time needed to perform a constant number of elementary bit operations), then:*

$$\Pr_{k,k',m}[\mathsf{D}_{k'}(\mathsf{E}_k(m)) = m] \leq \varepsilon.$$

PROOF. Consider the following adversary $\mathcal{A}$ against scheme Π: upon input 1^λ, adversary $\mathcal{A}$ chooses two messages m_0, m_1 uniformly in $\mathcal{M}$ and sends them to the challenger. When $\mathcal{A}$ receives the challenge ciphertext $c^* \leftarrow \mathsf{E}_k(m_b)$, it chooses a uniform key k' in $\mathcal{K}$ and computes $\mathsf{D}_{k'}(c^*) = m'$. If there exists $b' \in \{0, 1\}$ such that $m_{b'} = m'$, then $\mathcal{A}$ outputs b' otherwise $\mathcal{A}$ returns a uniform bit.
Let's analyze the success probability of $\mathcal{A}$: let F be the event that $m' = m_{b'}$ for a bit b'.

Since the k' as well as the secret key k and the message m_b chosen by the challenge are uniformly distributed, we have that:

$$\Pr[\mathsf{F}] = \Pr_{k,k',m_b}[\mathsf{D}_{k'}(\mathsf{E}_k(m_b)) = m_b].$$

If F happens, adversary $\mathcal{A}$ will output a correct guess of the bit b, this means that: $\mathsf{AdvPriv}^{\mathrm{CCA}}_{\mathcal{A},(\mathsf{E},\mathsf{D})}(\lambda) \geq \Pr[\mathsf{F}]$. Since we are assuming that (E,D) is a (t,ε)-OT-CCA secure encryption scheme we can conclude that:

$$\varepsilon \geq \mathsf{AdvPriv}^{\mathrm{CCA}}_{\mathcal{A},(\mathsf{E},\mathsf{D})}(\lambda) \geq \Pr_{k,k',m}[\mathsf{D}_{k'}(\mathsf{E}_k(m)) = m]. \qquad \square$$

REMARK 3.3.3. Notice that the adversary described in the proof above does not take advantage from a decryption oracle, so the same attack could also be performed to a OT-secure encryption scheme.

LEMMA 3.3.4. *If* Π *from construction 3.1.1 is a* $(Q'_d, Q'_H, t', \varepsilon')$*-CCA-secure scheme that uses a* $(Q''_d, Q''_H, t'', \varepsilon'')$*-CCA-secure symmetric scheme and let* H *be modeled as a random oracle. Then, the strong* n*-linear problem is* (Q_p, t, ε)*-hard relative to* Ggen*, where* $t \approx t', \quad \varepsilon \leq \varepsilon' + Q_p \cdot \varepsilon''$ *and* $Q_p = Q'_d$*.*

PROOF. Informally, the proof will be based on the following fact. Consider an adversary $\mathcal{A}$ that chooses $n+1$ elements $Y_1, \ldots, Y_n, Z$ and computes the value k of the hash function in these $n+1$ elements. Then, $\mathcal{A}$ chooses a message m and submits $(Y_1, \ldots, Y_n, \mathsf{E}_k(m))$ to the decryption oracle and receives back a plaintext $\bar{m}$. We can suppose that $m = \bar{m}$ if and only if $Z = Y_1^{x_1{}^{-1}} \cdots Y_n^{x_n{}^{-1}}$, where $(x_1, \ldots, x_n)$ is the secret key. This shows that the decryption oracle provides a way to simulate an oracle for the predicate n-lin.
To prove the theorem, we will use the games technique. Let $\mathcal{B}$ be an algorithm against the strong n-linear problem that runs in time at most t.
Game 0. Game 0 is the standard game against the strong n-linear problem defined in Section 2.3. In this game a challenger gives a tuple $(\mathscr{G}, g_1, \ldots, g_n, g_1^{x_1}, \ldots, g_1^{x_n})$ to $\mathcal{B}$. Adversary $\mathcal{B}$ has access to an oracle for the predicate $n\text{-}lin_{g,g_1,\ldots,g_n}$.
Game 1. In Game 1, we change the way in which we answer the oracle for predicate n-lin. In Game 1, after we computed the input $(\mathscr{G}, g_1, \ldots, g_n, g_1^{x_1}, \ldots, g_1^{x_n})$ for $\mathcal{B}$, we also choose a hash function H and use $(\mathscr{G}, g_1, \ldots, g_n, H)$ as public key for Π. In this way the experiment simulate internally the behavior of the encryption scheme. When $\mathcal{B}$ asks for the value of the predicate in $(Y_1, \ldots, Y_n, Z)$, we choose a message $m \leftarrow \mathcal{M}$ and compute $k = H(Y_1, \ldots, Y_n, Z)$. Then, we decrypt the ciphertext $(Y_1, \ldots, Y_n, \mathsf{E}_k(m))$. If the plaintext $\tilde{m}$ we get is equal to m, we returns 1 to $\mathcal{B}$ as value of $n\text{-}lin(Y_1, \ldots, Y_n, Z)$, otherwise we return 0.

Let G_i be the event that $\mathcal{B}$ wins in Game i and let F be the event we respond incorrectly at least one time to $\mathcal{B}$'s queries in Game 1. Since Game 1 and Game 0 proceed identically until F happens, by difference Lemma 1.5.1 we have:

$$|\Pr[G_1(\lambda) = 1] - \Pr[G_0(\lambda) = 1]| \leq \Pr[\mathsf{F}].$$

Let $\mathsf{F_i}$ be the event that we give an incorrect answer to the i-th query, then:

$$\Pr[\mathsf{F}] = \Pr[\mathsf{F}_1 \vee \cdots \vee \mathsf{F}_{Q_p}].$$

Notice that if in Game 0 the answer to the predicate n-lin on input $(Y_1, \ldots, Y_n, Z)$ is 1, then by correctness of Π, the answer is 1 in Game 1, as well. Vice versa, if the answer in Game 0 is 0, the value $k = H(Y_1, \ldots, Y_n, Z)$ and the key used to decrypt the message in

Game 1 are independently distributed in $\mathcal{B}$'s view, so by Lemma 3.3.2, the probability that the answer in Game 1 is not 0 is bounded by ε''. Thus $\Pr[\mathsf{F}_i] \leq \varepsilon''$. So:

$$\Pr[\mathsf{F}] \leq Q_p \cdot \varepsilon''.$$

To conclude the proof, we will build an adversary $\mathcal{A}$ against scheme Π such that:

$$\mathsf{AdvPub}^{\mathrm{CCA}}_{\mathcal{A},\Pi}(\lambda) \geq |\Pr[G_1(\lambda) = 1] - 1/2|\,.$$

Upon input $pk = (\mathscr{G}, X_1, \ldots, X_n, H)$, adversary $\mathcal{A}$ chooses two messages m_0, m_1 in the message space $\mathcal{M}$ and returns them to the challenger. When the challenger gives $\mathcal{A}$ the challenge ciphertext $(Y_1^*, \ldots, Y_n^*, c^*)$, adversary $\mathcal{A}$ invokes $\mathcal{B}$ on input $(\mathscr{G}, X_1, \ldots, X_n, Y_1^*, \ldots, Y_n^*)$. At this point, adversary $\mathcal{B}$ can ask the value of the predicate *n-lin* in $(Y_1, \ldots, Y_n, Z)$. In order to process this query, adversary $\mathcal{A}$ chooses a message m in $\mathcal{M}$ and computes $k = H(Y_1, \ldots, Y_n, Z)$. Then, $\mathcal{A}$ sends the ciphertext $(Y_1, \ldots, Y_n, \mathsf{E}_k(m))$ to the challenger for decryption. If the plaintext $\tilde{m}$ output by the challenger is equal to m, adversary $\mathcal{A}$ returns 1 to $\mathcal{B}$ as value of *n-lin*$(Y_1, \ldots, Y_n, Z)$, otherwise $\mathcal{A}$ returns 0. When $\mathcal{B}$ returns an element h, adversary $\mathcal{A}$ computes $k^* = H(Y_1^*, \ldots, Y_n^*, h)$ and $m^* = \mathsf{D}_{k^*}(c^*)$. If there exists $b' \in \{0,1\}$ such that $m_b = m^*$, adversary $\mathcal{A}$ outputs b', otherwise it returns fail.

We notice that the simulation provided by $\mathcal{A}$ of Game 1 is perfect and the running time of the two algorithms $\mathcal{A}$ and $\mathcal{B}$ is approximately the same. Moreover, if $\mathcal{B}$ wins Game 1, adversary $\mathcal{A}$ will be able to decrypt correctly the challenge ciphertext. This means that:

$$\mathsf{AdvPub}^{\mathrm{CCA}}_{\mathcal{A},\Pi}(\lambda) \geq |\Pr[G_1(\lambda) = 1] - 1/2|\,.$$

We can conclude that:

$$\mathsf{Adv}^{Sn\text{-}lin}_{\mathcal{B},\mathsf{Ggen}}(\lambda) \leq \mathsf{AdvPub}^{\mathrm{CCA}}_{\mathcal{A},\Pi}(\lambda) + Q_p \cdot \varepsilon''. \tag{3.3}$$

Since, by hypothesis, Π is a $(Q'_d, Q'_H, t', \varepsilon')$-CCA-secure encryption scheme, equation (3.3) becomes:

$$\mathsf{Adv}^{Sn\text{-}lin}_{\mathcal{B},\mathsf{Ggen}}(\lambda) \leq \varepsilon' + Q_p \cdot \varepsilon''.$$

So, we can conclude that the strong n-linear problem is (t, ε)-hard relative to Ggen, where $\varepsilon \leq \varepsilon' + Q_p \cdot \varepsilon''$ and $t \approx t'$. Also, $Q_p = Q'_d$ since adversary $\mathcal{A}$ invokes the decryption oracle when B invokes the oracle for the predicate $n - lin$. □

LEMMA 3.3.5. *Suppose that* (E, D) *is a* $(Q'_d, Q'_H, t', \varepsilon')$*-OT-CCA-secure symmetric encryption scheme, that* H *is modeled as a random oracle and that the strong* n*-linear DH problem is* $(Q''_p, t'', \varepsilon'')$*-hard relative to* Ggen*. Then* Π *from construction 3.1.1 is* $(Q_d, Q_H, t, \varepsilon)$*-secure against chosen ciphertext attacks, where* $t \approx t' \approx t''$*,* $\varepsilon \leq \varepsilon' + \varepsilon'' + \frac{Q_d}{2^{n\lambda}}$ *and* $Q''_p \leq Q_H = Q'_H$ *and* $Q'_d \leq Q_d$*.*

PROOF. The idea behind this proof is that the predicate *n-lin* is needed to decide if we are facing a "correct" or an "incorrect" instance of the n-linear problem during the reduction to the strong n-linear problem. In fact, at this point, we will have to simulate the decryption oracle without having the secret key. In particular, suppose we have to decrypt a ciphertext $(Y_1, \ldots, Y_n, c)$. We can check in the list of the random oracle if we have a tuple $(Y_1, \ldots, Y_n, Z, k)$, but we need also to check if Z is the right solution, otherwise the adversary could detect our incorrect decryption using the random oracle.
We take an adversary $\mathcal{A}$ against scheme Π running in time t and we keep it fixed in all the following games.
Game 0. The first game is the standard game for CCA-security defined in 1.2.2. Let us recall it briefly. At the beginning of the game, after having computed the public and the

secret key $(pk, sk) = \big((\mathscr{G}, X_1 \ldots, X_n, H), (x_1, \ldots, x_n)\big)$, the challenger chooses $y_1, \ldots, y_n$ in $\mathbb{Z}_q$ uniformly and computes $(Y_1^*, \ldots, Y_n^*) = (X_1^{y_1}, \ldots, X_n^{y_n})$ and $Z^* = g^{y_1+\cdots+y_n}$. Then, the challenger computes $k^* = H(Y_1^*, \ldots, Y_n^*, Z^*)$. When $\mathcal{A}$ asks for the decryption of a ciphertext, the challenger uses the secret key sk to decrypt the message and gives back to $\mathcal{A}$ the corresponding plaintext. When $\mathcal{A}$ outputs two messages m_0, m_1, the challenger gives $\mathcal{A}$ the challenge ciphertext $(Y_1^*, \ldots, Y_n^*, \mathsf{E}_{k^*}(m_b))$. If $\mathcal{A}$ invokes again the decryption oracle, the challenger decrypts as above. Finally $\mathcal{A}$ outputs its guess b'. We recall that $\mathcal{A}$ has access to an oracle for H throughout the game.

Game 1. Game 1 works exactly like Game 0, except in the following aspect: the challenger aborts and returns $\perp$ if, before having seen the challenge ciphertext, $\mathcal{A}$ asks for the decryption of $(Y_1^*, \ldots, Y_n^*, c)$, for some ciphertext c.

Let G_i be the event that $\mathcal{A}$ wins Game i and let L be the following event: adversary $\mathcal{A}$ asks for the decryption of $(Y_1^*, \ldots, Y_n^*, c)$, for some c, before having seen the challenge ciphertext. Then, Game 0 and Game 1 are identical until L happens. By the difference lemma 1.5.1, we have that:

$$|\Pr[G_0(\lambda) = 1] - \Pr[G_1(\lambda) = 1]| \leq \Pr[\mathsf{L}].$$

Since the values $(Y_1^*, \ldots, Y_n^*)$ are information theoretically hidden from $\mathcal{A}$'a view until it sees the challenge ciphertext, the probability that $\mathcal{A}$ queries them is $1/q^n$. So, let L_i be the event that $\mathcal{A}$ asks for the decryption of $((Y_1^*, \ldots, Y_n^*, c))$ at the i-th query, then:

$$\Pr[\mathsf{L}] = \Pr[\mathsf{L}_1 \vee \cdots \vee \mathsf{L}_{Q_d}] \leq \frac{1}{q^n} + \cdots + \frac{1}{q^n} = \frac{Q_d}{q^n}. \tag{3.4}$$

Hence, we can conclude that $\Pr[\mathsf{L}]$ is negligible.

Game 2. We define the Game 2 to be as Game 1 except in the following two aspects: when the challenger has to process the challenge ciphertext, it chooses a uniform element k in $\{0,1\}^\ell$ and outputs $(Y_1^*, \ldots, Y_n^*, \mathsf{E}_k(m_b))$. Secondly, the challenger will use this k to process the decryption queries of the form $(Y_1^*, \ldots, Y_n^*, c)$, for any c.

Consider the following event F: adversary $\mathcal{A}$ asks for $H(Y_1^*, \ldots, Y_n^*, \prod_{i=1}^n (Y_i^*)^{x_i^{-1}})$. Then Game 1 and Game 2 proceed identically until F occurs. By the lemma 1.5.1, we have that:

$$|\Pr[G_2(\lambda) = 1] - \Pr[G_1(\lambda) = 1]| \leq \Pr[\mathsf{F}].$$

Claim: The probability that F occurs is bounded by ε''.

We will prove that there exists an adversary $\mathcal{B}$ against the strong n-linear problem that runs in approximately the same time of $\mathcal{A}$ such that:

$$\Pr[\mathsf{F}] \leq \mathsf{Adv}^{Sn\text{-}lin}_{\mathcal{B},\mathsf{Ggen}}(\lambda).$$

Consider the following adversary $\mathcal{B}$ against the strong n-linear DH problem: when $\mathcal{B}$ receives an input $((\mathbb{G}, q, g), g_1, \ldots, g_n, g_1^{x_1}, \ldots, g_n^{x_n})$ it chooses a hash function $H : \{0,1\}^* \to \{0,1\}^\ell$. Then, it samples k^* in $\{0,1\}^\ell$ and sets $H(g_1^{x_1} \ldots, g_n^{x_n}, g^{x_1+\cdots+x_n}) = k^*$, even if it does not know the value $g^{x_1+\cdots+x_n}$. Then, $\mathcal{B}$ creates a table T whose entries can be of two types:

- Explicit type: $(g_1^{y_1}, \ldots, g_n^{y_n}, Z, k, p) \in T$ means that $k = H(g_1^{y_1}, \ldots, g_n^{y_n}, Z)$ and p is the value given by the predicate $n\text{-}lin$: $p = n\text{-}lin(g_1^{y_1}, \ldots, g_n^{y_n}, Z) \in \{0,1\}$.
- Implicit type: $(g_1^{y_1}, \ldots, g_n^{y_n}, *, k, 1) \in T$ means that $k = H(g_1^{y_1}, \ldots, g_n^{y_n}, g^{y_1+\cdots+y_n})$

After this, adversary $\mathcal{B}$ initializes the table T with the value $(g_1^{x_1}, \ldots, g_n^{x_n}, *, k^*, 1)$ and invokes $\mathcal{A}$ on input $pk = (\mathscr{G}, g_1, \ldots, g_n, H)$. When $\mathcal{A}$ asks for $H(g_1^{y_1}, \ldots, g_n^{y_n}, Z)$, adversary $\mathcal{B}$ answers as follows:

- If there exists an entry $(g_1^{y_1}, \ldots, g_n^{y_n}, Z, k, p)$ in T, adversary $\mathcal{B}$ returns k.

- If there exists an entry $(g_1^{y_1}, \ldots, g_n^{y_n}, *, k, 1)$ in T and $n\text{-}lin(g_1^{y_1}, \ldots, g_n^{y_n}, Z) = 1$, $\mathcal{B}$ returns k and replaces $(g_1^{y_1}, \ldots, g_n^{y_n}, *, k, 1)$ with $(g_1^{y_1}, \ldots, g_n^{y_n}, Z, k, 1)$ in T.
- Otherwise $\mathcal{B}$ chooses $k \leftarrow \mathbb{Z}_q$, returns k and stores $(g_1^{y_1}, \ldots, g_n^{y_n}, Z, k, p)$ in T

When $\mathcal{A}$ asks for the decryption of $(g_1^{y_1}, \ldots, g_n^{y_n}, c)$ then $\mathcal{B}$ answers as follows:

- If there exists an entry $(g_1^{y_1}, \ldots, g_n^{y_n}, Z, k, 1)$ in T, $\mathcal{B}$ returns $m = \mathsf{D}_k(c)$.
- If there exists an entry $(g_1^{y_1}, \ldots, g_n^{y_n}, *, k, 1)$ in T, $\mathcal{B}$ returns $m = \mathsf{D}_k(c)$.
- Otherwise $\mathcal{B}$ chooses $k \leftarrow \{0,1\}^\ell$, stores $(g_1^{y_1}, \ldots, g_n^{y_n}, *, k, 1)$ in T and returns $m = \mathsf{D}_k(c)$.

When $\mathcal{A}$ outputs two messages m_0, m_1, adversary $\mathcal{B}$ chooses a bit $b \leftarrow \{0,1\}$ and gives $(g_1^{x_1}, \ldots, g_n^{x_n}, \mathsf{E}_{k^*}(m_b))$ as challenge ciphertext to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', adversary $\mathcal{B}$ ignores it and searches in T if there exists an entry of the type $(g_1^{x_1}, \ldots, g_n^{x_n}, Z, k^*, 1)$. In this case $\mathcal{B}$ outputs Z, otherwise returns fail.
We notice that the simulation that $\mathcal{B}$ provides to $\mathcal{A}$ is perfect and that the running times of $\mathcal{A}$ and $\mathcal{B}$ are approximately the same. Moreover, as soon as event F happens, the algorithm $\mathcal{B}$ will be able to solve correctly its instance of the strong n-linear problem. This means that:

$$\mathsf{Adv}^{Sn\text{-}lin}_{\mathcal{B},\mathsf{Ggen}}(\lambda) \geq \Pr[\mathsf{F}].$$

Since we are assuming that the strong n-linear problem is (t'', ε'')-hard, we can conclude that:

$$\Pr[\mathsf{F}] \leq \varepsilon''. \tag{3.5}$$

We notice that adversary $\mathcal{B}$ asks oracle queries for the predicate $n\text{-}lin$ only when adversary $\mathcal{A}$ queries the random oracle. So, $Q_p'' \leq Q_H$.

To conclude the proof we will show that the advantage of $\mathcal{A}$ in Game 2 is bounded by ε'.
Claim: There exists an adversary $\mathcal{A}'$ against the symmetric scheme (E, D) that runs in about the same time of $\mathcal{A}$ such that:

$$\Pr[G_2(\lambda) = 1] \leq \frac{1}{2} + \mathsf{AdvPriv}^{\mathrm{CCA}}_{\mathcal{A}',(\mathsf{E},\mathsf{D})}(\lambda).$$

To conclude the claim we build an adversary $\mathcal{A}'$ against the CCA-security of (E, D) as follows: upon input 1^λ, adversary $\mathcal{A}'$ computes a pair of keys

$$(pk, sk) = ((\mathscr{G}, g^{x_1}, \ldots, g^{x_n}, H), (x_1, \ldots, x_n)) \leftarrow \mathsf{Gen}(1^\lambda).$$

Next, $\mathcal{A}'$ chooses $y_1, \ldots, y_n \leftarrow \mathbb{Z}_q$ and computes $Y_i^* = g^{x_i y_i}$, $Z^* = g^{y_1 + \cdots + y_n}$. Adversary $\mathcal{A}'$ sets $k^* = H(Y_1^*, \ldots, Y_n^*, Z^*)$. Next, $\mathcal{A}'$ invokes $\mathcal{A}$ on input pk. When $\mathcal{A}$ asks for $H(Y_1, \ldots, Y_n, Z)$, adversary $\mathcal{A}$ simulates the random oracle as usual. When $\mathcal{A}$ asks for the decryption of $(Y_1, \ldots, Y_n, c)$, adversary $\mathcal{A}'$ answers as follows:

- If $(Y_1, \ldots, Y_n) = (Y_1^*, \ldots, Y_n^*)$ then $\mathcal{A}'$ aborts and returns $\perp$.
- Otherwise sets $Z = Y_1^{x_1^{-1}} \cdots Y_n^{x_n^{-1}}$, computes $k = H(Y_1, \ldots, Y_n, Z)$ and returns $m = \mathsf{D}_k(c)$.

When $\mathcal{A}$ outputs two messages m_0, m_1, adversary $\mathcal{A}'$ gives them to the challenger and receives the challenger ciphertext c^*. Then, $\mathcal{A}'$ gives $(Y_1^*, \ldots, Y_n^*, c^*)$ to $\mathcal{A}$ as challenger ciphertext. When $\mathcal{A}$ asks for more values of H, adversary $\mathcal{A}'$ processes the queries as above, when $\mathcal{A}'$ asks for more decryptions of the type $(Y_1, \ldots, Y_n, c)$, adversary $\mathcal{A}$ does the following:

- If $(Y_1, \ldots, Y_n) = (Y_1^*, \ldots, Y_n^*)$ then $\mathcal{A}'$ gives c to the challenger and gives back the result to $\mathcal{A}$.
- Otherwise $\mathcal{A}'$ sets $Z = Y_1^{x_1^{-1}} \cdots Y_n^{x_n^{-1}}$, computes $k = H(Y_1, \ldots, Y_n, Z)$ and returns $m = \mathsf{D}_k(c)$.

Finally, $\mathcal{A}'$ outputs whatever $\mathcal{A}$ outputs.

Every time $\mathcal{A}$ wins Game 2, adversary $\mathcal{A}'$ is able to answer correctly to the experiment of CCA-security. Moreover, the simulation of Game 2 provided by $\mathcal{A}'$ to $\mathcal{A}$ is perfect. This means that

$$\mathsf{AdvPriv}^{\mathrm{CCA}}_{\mathcal{A}',(\mathsf{E},\mathsf{D})}(\lambda) \geq \left|\Pr[G_2(\lambda) = 1] - \frac{1}{2}\right|.$$

We notice that the running time of $\mathcal{A}$ and $\mathcal{A}'$ are approximately the same. Hence, since we are assuming that (E, D) is a $(Q'_d, Q'_H, t', \varepsilon')$-OT-CCA secure encryption scheme, we can conclude that:

$$\left|\Pr[G_2(\lambda) = 1] - \frac{1}{2}\right| \leq \varepsilon'. \tag{3.6}$$

We notice that adversary $\mathcal{A}'$ queries the random oracle every time $\mathcal{A}$ does and $\mathcal{A}'$ queries the decryption oracle only if $\mathcal{A}$ sends some special ciphertext to the decryption oracle. This means that $Q'_H = Q_H$ and $Q'_d \leq Q_d$.

Collecting all the pieces from equations (3.4), (3.5) and (3.6), we have that scheme Π is a $(Q_d, Q_H, t, \varepsilon)$-CCA-secure scheme, where

$$\varepsilon \leq \varepsilon' + \varepsilon'' + \frac{Q_d}{q^n}. \qquad \square$$

CHAPTER 4

A Scheme Based on the Linear Problems

In the previous chapter we proved that scheme Π is CCA-secure if and only if the strong n-linear problem is hard. However, it would be more interesting to have a scheme based on some weaker and non-interactive problem. In particular, we can wonder whether the hardness of the n-linear problem is sufficient to obtain CCA-security. We will see in this chapter that, by introducing a new problem, we can indeed construct a CCA-secure scheme whose security is based on the hardness of the n-linear problem.

4.1. The Twin Strong Linear Problems

The problems that we are going to define generalize the twin Diffie-Hellman problem given by Cash et al. in [CKS08].

Let $\mathscr{G} = (\mathbb{G}, q, g)$ be a group description. Given $\mathscr{G}, g_1, \ldots, g_n, g_1^{x_1}, \ldots, g_n^{x_n}$ and $g_1^{y_1}, \ldots, g_n^{y_n}$ with $x_i, y_i \in \mathbb{Z}_q$, the twin n-linear problem consists in computing the element $g^{x_1+\cdots+x_n} \in \mathbb{G}$. As we explained in Chapter 2, n can be a polynomial in the security parameter λ.

We are now ready to define the problem that we need. In fact, as we have noticed in the previous chapter, we will also need the strong version of the twin linear problem.

Definition 4.1.1. Let $g, g_1, \ldots, g_n, g_1^{x_1}, \ldots, g_n^{x_n}, g_1^{y_1}, \ldots, g_n^{y_n}$ be elements of $\mathbb{G}$. The predicate

$$n\text{-}2lin_{g,g_1,\ldots,g_n,g_1^{y_1},\ldots,g_n^{y_n}}(g_1^{x_1}, \ldots, g_n^{x_n}, Z_1, Z_2)$$

returns 1 if and only if $Z_1 = g^{x_1+\cdots+x_n}$ and $Z_2 = g_1^{x_1 y_1} \cdots g_n^{x_n y_n}$.

To an algorithm $\mathcal{A}$ and a group generation algorithm Gen, we associate the following experiment $\mathsf{Exp}^{Sn\text{-}2lin}_{\mathcal{A},\mathsf{Gen}}(\lambda)$:

1. The challenger chooses a group description $\mathscr{G} = (\mathbb{G}, q, g) \leftarrow \mathsf{Gen}(1^\lambda)$ and n uniform elements $g_1, \ldots, g_n$ in $\mathbb{G}$. The challenger also chooses $r_1, \ldots, r_n, s_1, \ldots, s_n \leftarrow \mathbb{Z}_q$.
2. $\mathcal{A}$ is given $\mathscr{G}, g_1, \ldots, g_n, g_1^{r_1}, \ldots, g_n^{r_n}, g_1^{s_1}, \ldots, g_n^{s_n}$.
3. $\mathcal{A}$ has access to an oracle for $n\text{-}2lin_{g,g_1,\ldots,g_n,g_1^{s_1},\ldots,g_n^{s_n}}$.
4. $\mathcal{A}$ outputs $h \in \mathbb{G}$.

The output of the experiment is 1 if and only if $h = g^{r_1+\cdots+r_n}$ and 0 otherwise. We define $\mathsf{Adv}^{Sn\text{-}2lin}_{\mathcal{A},\mathsf{Gen}}(\lambda) = \Pr[\mathsf{Exp}^{Sn\text{-}2lin}_{\mathcal{A},\mathsf{Gen}}(\lambda) = 1]$.

Definition 4.1.2. The strong twin n-linear problem is (Q, t, ε)-**hard** relative to Gen if, for every algorithm $\mathcal{A}$ that runs in time bounded by t and that makes at most Q queries to the oracle for the predicate $n\text{-}2lin$, the advantage $\mathsf{Adv}^{Sn\text{-}2lin}_{\mathcal{A},\mathsf{Gen}}$ is bounded by the positive constant ε.

4.2. Equivalence

Our purpose is showing that the strong twin n-linear problem is tightly equivalent to the n-linear problem in the concrete setting. The hardest part consists in proving that the hardness of the n-linear problem implies the hardness of strong twin n-linear problem. In fact, to do this, we will have to simulate the oracle for the n-2lin predicate without knowing the corresponding discrete logarithms. The following Lemma 4.2.1, which is a generalization of the trapdoor lemma given in [CKS08], will help us to simulate the predicate.

LEMMA 4.2.1. *Let $\mathbb{G}$ be a cyclic group of order a prime number q and g a generator of $\mathbb{G}$. Let $g_1, \dots g_n, s_1, \dots, s_n, t$ be mutually independent random variables, where $g_1, \dots, g_n$ take value in $\mathbb{G} \setminus \{Id\}$ and $s_1, \dots, s_n, t$ are uniformly distributed over $\mathbb{Z}_q$. Define n new random variables u_i to be $g_i^{s_i} g^t$ for $i \in \{1, \dots, n\}$. Finally, let $\hat{c}_1, \dots, \hat{c}_n, \hat{d}_1, \hat{d}_2$ be random variables defined as some function of the g_i's and of the u_i's. Then, if $u_i = g_i^{y_i}$ and $g = g_i^{x_i}$, the probability that the truth value of*

$$\hat{c}_1^{s_1} \cdots \hat{c}_n^{s_n} \hat{d}_1^t = \hat{d}_2 \tag{4.1}$$

does not agree with the truth value of

$$\hat{c}_1^{x_1} \cdots \hat{c}_n^{x_n} = \hat{d}_1 \ \wedge \ \hat{c}_1^{y_1} \cdots \hat{c}_n^{y_n} = \hat{d}_2 \tag{4.2}$$

is at most $\frac{1}{q}$.

PROOF. Let R be the probability space defined by the random variables $g_1, \dots, g_n, s_1$, $s_2 \dots, s_n, t$. Now, we fix the values $g_1, \dots, g_n$ and $u_1, \dots, u_n$. The resulting conditioned probability space $R_{g_1,\dots,g_n,u_1,\dots,u_n}$ can be expressed by picking a uniform random variable $t \leftarrow \mathbb{Z}_q$, since the system of equations:

$$\begin{cases} u_1 &= g_1^{s_1} g^t \\ \vdots & \vdots \\ u_n &= g_n^{s_n} g^t \end{cases}$$

allows us to determine the variables $s_1, \dots, s_n$ as functions of t, while there are no conditions on t itself.
Notice that, for every i, we have $y_i = s_i + t x_i$. In fact:

$$u_i = g_i^{s_i} g^t = g_i^{s_i} (g_i^{x_i})^t = g_i^{s_i + t x_i} = g_i^{y_i}$$

which implies $y_i = s_i + t x_i$.
Now, assume that (4.2) holds. By substituting the first equation of (4.2) into (4.1), we get that:

$$\hat{c}_1^{s_1} \cdots \hat{c}_n^{s_n} \hat{d}_1^t = \hat{d}_2$$

is equivalent to

$$\hat{c}_1^{s_1} \cdots \hat{c}_n^{s_n} (\hat{c}_1^{x_1} \cdots \hat{c}_n^{x_n})^t = \hat{d}_2$$

Then, the latter equation is equivalent to:

$$\hat{c}_1^{s_1 + t x_1} \cdots \hat{c}_n^{s_n + t x_n} = \hat{d}_2.$$

By the relations $y_i = s_i + t x_i$, equation 4.2 becomes:

$$\hat{c}_1^{y_1} \cdots \hat{c}_n^{y_n} = \hat{d}_2.$$

Thus, by the second equation of (4.2), we can conclude that (4.2) implies (4.1).

Now, observe that (4.1) can be written as:

$$\frac{(\hat{c}_1^{s_1}\cdots\hat{c}_n^{s_n})(\hat{c}_1^{tx_1}\cdots\hat{c}_n^{tx_n})}{\hat{c}_1^{tx_1}\cdots\hat{c}_n^{tx_n}}\hat{d}_1^t = \hat{d}_2. \tag{4.3}$$

Since $y_i = s_i + tx_i$, equation (4.3) is exactly the same as:

$$\frac{\hat{c}_1^{y_1}\cdots\hat{c}_n^{y_n}}{\hat{d}_2} = \left(\frac{\hat{c}_1^{x_1}\cdots\hat{c}_n^{x_n}}{\hat{d}_1}\right)^t. \tag{4.4}$$

If we assume that $\hat{c}_1^{x_1}\cdots\hat{c}_n^{x_n} = \hat{d}_1$ but $\hat{c}_1^{y_1}\cdots\hat{c}_n^{y_n} \neq \hat{d}_2$, then (4.4) clearly does not hold. It remains to consider the case $\hat{c}_1^{x_1}\cdots\hat{c}_n^{x_n} \neq \hat{d}_1$. Then, the right hand side of (4.4) is a uniform element in $\mathbb{G}$ (since t is a uniform variable) and the left hand side is a fixed element of the group. Thus, the probability that (4.4) holds in this case is at most $\frac{1}{q}$. So (4.1) agrees with (4.2) except with probability $\frac{1}{q}$.

□

We will prove now that the strong twin n-linear and the n-linear problems are tightly equivalent. We will start by proving the easy direction, namely that the hardness of the strong twin n-linear problem implies the hardness of the n-linear problem.

LEMMA 4.2.2. *If the strong twin n-linear problem is (t,ε)-hard relative to* Ggen*, then also the n-linear problem is (t,ε)-hard relative to* Ggen*.*

PROOF. Assume the strong twin n-linear problem is (t,ε)-hard. Let $\mathcal{A}'$ be an adversary solving the n-linear problem in time t; we build an adversary $\mathcal{B}'$ solving the strong twin n-linear problem as follows: adversary $\mathcal{B}'$ is given $(\mathscr{G}, g_1, \ldots, g_n, g_1^{r_1}, \ldots, g_n^{r_n}, g_1^{s_1}, \ldots, g_n^{s_n})$ as input. Then, adversary $\mathcal{B}'$ invokes $\mathcal{A}'$ on input $(\mathscr{G}, g_1, \ldots, g_n, g_1^{r_1}, \ldots, g_n^{r_n})$ and return any Z_1 output by $\mathcal{A}'$.
We notice that, every time $\mathcal{A}'$ solves correctly its instance of the n-linear problem, adversary $\mathcal{B}$ outputs a correct solution of the strong twin n-linear problem, so:$\mathsf{Adv}^{Sn\text{-}2lin}_{\mathcal{B}',\mathsf{Ggen}}(\lambda) \geq \mathsf{Adv}^{n\text{-}lin}_{\mathcal{A}',\mathsf{Ggen}}(\lambda)$. Since the running time of the two algorithms are the same, and we are assuming that the strong twin n-linear problem is (t,ε)-hard, we have that:

$$\varepsilon \geq \mathsf{Adv}^{n\text{-}lin}_{\mathcal{A}',\mathsf{Ggen}}(\lambda).$$

This proves that n-linear problem is (t,ε)-hard. □

To conclude the proof of the equivalence we will show that the hardness of the n-linear problem implies the hardness of strong twin n-linear problem. To do this we will use the Trapdoor Lemma 4.2.1.

LEMMA 4.2.3. *If the n-linear problem is (t',ε')-hard relative to* Ggen*, then the strong twin n-linear problem is (Q,t,ε)-hard relative to* Ggen*, where $t \approx t'$ and $\varepsilon \leq \varepsilon' + Q/2^\lambda$.*

PROOF. To prove this lemma we will use the games technique.
Assume the n-linear problem is (t,ε)-hard and let $\mathcal{B}$ be a strong twin n-linear DH adversary running in time t.
Game 0. In Game 0, adversary $\mathcal{B}$ plays the standard game we defined in Section 4.1.
Game 1. In Game 1 we change the way we answer to decisional queries of $\mathcal{B}$. When we compute the input $(\mathscr{G}, g_1, \ldots, g_n, g_1^{r_1}, \ldots, g_n^{r_n}, u_1, \ldots, u_n)$ of $\mathcal{B}$, we also choose $t \leftarrow \mathbb{Z}_q$ and compute $s_1, \ldots, s_n \in \mathbb{Z}_q$ such that $u_i = g_i^{s_i} g^t$ for $i \in \{1, \ldots, n\}$. When $\mathcal{B}$ asks for $n\text{-}2lin(\hat{c}_1, \ldots, \hat{c}_n, \hat{d}_1, \hat{d}_2)$, we process the decision query by checking if $\hat{c}_1^{s_1}\cdots\hat{c}_n^{s_n}\hat{d}_1^t = \hat{d}_2$ holds.

Now, let G_i be the event that $\mathcal{B}$ wins Game i and let F be the event that we answer incorrectly to the queries of $\mathcal{B}$ at least one time. Then, by Difference Lemma 1.5.1:

$$|\Pr[G_1(\lambda) = 1] - \Pr[G_0(\lambda) = 1]| \leq \Pr[\mathsf{F}].$$

Let F_i be the event that we respond incorrectly at the i-th query, then $\Pr[\mathsf{F}] = \Pr[\mathsf{F}_1 \vee \cdots \vee \mathsf{F}_Q]$. By Lemma 4.2.1, we have $\Pr[\mathsf{F}_i] \leq 1/q$, hence:

$$\Pr[F] = \Pr[\mathsf{F}_1 \vee \cdots \vee \mathsf{F}_Q] \leq \sum_{i=1}^{Q} \Pr[\mathsf{F}_i] \leq \frac{Q}{q}.$$

Thus, we have:

$$|\Pr[G_1(\lambda) = 1] - \Pr[G_0(\lambda) = 1]| = |\Pr[G_1(\lambda) = 1] - \mathsf{Adv}^{Sn\text{-}lin}_{\mathcal{B},\mathsf{Ggen}}(\lambda)| \leq \frac{Q}{q}. \tag{4.5}$$

Now, if $\mathsf{Adv}^{Sn\text{-}lin}_{\mathcal{B},\mathsf{Ggen}}(\lambda) \geq \Pr[G_1(\lambda) = 1]$, equation (4.5) becomes:

$$\mathsf{Adv}^{Sn\text{-}lin}_{\mathcal{B},\mathsf{Ggen}}(\lambda) \leq \Pr[G_1(\lambda) = 1] + \frac{Q}{q}. \tag{4.6}$$

Otherwise, if $\mathsf{Adv}^{Sn\text{-}lin}_{\mathcal{B},\mathsf{Ggen}}(\lambda) \leq \Pr[G_1(\lambda) = 1]$, we can conclude also that:

$$\mathsf{Adv}^{Sn\text{-}lin}_{\mathcal{B},\mathsf{Ggen}}(\lambda) \leq \Pr[G_1(\lambda) = 1] \leq \Pr[G_1(\lambda) = 1] + \frac{Q}{q}.$$

To bound the probability that $\mathcal{B}$ wins game 1 we build an adversary $\mathcal{A}$ against the n-linear problem. Adversary $\mathcal{A}$ is constructed as follows: on input $(\mathscr{G}, g_1, \ldots, g_n, g_1^{r_1}, \ldots, g_n^{r_n})$, $\mathcal{A}$ chooses $s_1, \ldots, s_n, t \leftarrow \mathbb{Z}_q$ and computes $u_i = g_i^{s_i} g^t$ for $i \in \{1, \ldots, n\}$, then it invokes $\mathcal{B}$ on input $(\mathscr{G}, g_1, \ldots, g_n, g_1^{r_1}, \ldots, g_n^{r_n}, u_1, \ldots, u_n)$. When $\mathcal{B}$ asks for $n\text{-}lin(\hat{c}_1, \ldots, \hat{c}_n, \hat{d}_1, \hat{d}_2)$, adversary $\mathcal{A}$ processes the decision query by checking if $\hat{c}_1^{s_1} \cdots \hat{c}_n^{s_n} \hat{d}_1^t = \hat{d}_2$ holds. Finally, adversary $\mathcal{A}$ outputs whatever $\mathcal{B}$ outputs.

We notice that the input received by $\mathcal{B}$ from $\mathcal{A}$ is identically distributed to $\mathcal{B}$'s input in Game 1. Moreover, if $\mathcal{B}$ wins Game 1, adversary $\mathcal{A}$ solves correctly its instance of the n-linear problem, so $\mathsf{Adv}^{n\text{-}lin}_{\mathcal{A},\mathsf{Ggen}}(\lambda) \geq \Pr[G_1(\lambda) = 1]$. Thus, by equation (4.6), we have:

$$\mathsf{Adv}^{Sn\text{-}lin}_{\mathcal{B},\mathsf{Ggen}}(\lambda) \leq \mathsf{Adv}^{n\text{-}lin}_{\mathcal{A},\mathsf{Ggen}}(\lambda) + \frac{Q}{q}.$$

Since we are assuming that the n-linear problem is (t', ε')-hard and since the running time of the two algorithms $\mathcal{A}$ and $\mathcal{B}$ are approximately the same, the strong twin n-linear problem is (Q, t, ε)-hard, where

$$\varepsilon \leq \varepsilon' + \frac{Q}{q}. \qquad \square$$

Corollary 4.2.4. *The strong twin n-linear DH problem is tightly equivalent to the n-linear DH problem.*

Remark 4.2.5. We could have given a different definition of the strong twin DH problem, that probably would have looked more reasonable. In particular, one could wonder why in the predicate we check that $Z_2 = g^{y_1 s_2} \cdots g^{y_n s_n}$ but the final output of the problem is just the element $g^{r_1 + \cdots + r_n}$. We could as well have defined the final output to be the pair $(g^{r_1 + \cdots + r_n}, g^{r_1 s_2} \cdots g^{r_n s_n})$ and nothing would have changed. The point is that we would

not have had a tight reduction in Lemma 4.2.2 with this latter symmetric definition. More precisely, with the new definition we can only prove that:

$$\mathsf{Adv}^{n\text{-}lin}_{\mathcal{A}',\mathsf{Ggen}}(\lambda) \approx (\mathsf{Adv}^{Sn\text{-}2lin}_{\mathcal{B}',\mathsf{Ggen}}(\lambda))^{n+1}.$$

4.3. Second Scheme

In this section we will give the construction of a scheme that is CCA-secure under the hardness of the strong twin n-linear problem. This new scheme is a modification of scheme Π given in Construction 4.3.1.

Construction 4.3.1. Let Ggen be a group-generation algorithm. Let (E,D) be a symmetric encryption scheme with key space $\{0,1\}^\ell$ and with message space $\mathcal{M}$. We define a new public-key encryption scheme $\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ with message space $\mathcal{M}$ as follows:

- The key-generation algorithm Gen', upon input 1^λ, generates a group description $\mathscr{G} = (\mathbb{G}, q, g) \leftarrow \mathsf{Ggen}(1^\lambda)$ and a hash function $H : \{0,1\}^* \to \{0,1\}^\ell$. Then it samples $2n$ uniform elements $x_1,\ldots,x_n,v_1,\ldots,v_n$ in $\mathbb{Z}_q^*$ and outputs
$$(sk, pk) = \big((x_1,\ldots,x_n,v_1,\ldots,v_n),(\mathscr{G},X_1,\ldots,X_n,V_1,\ldots,V_n,H)\big)$$
where $X_i = g^{x_i}$ and $V_i = X_i^{v_i}$ for every i in $\{1,\ldots,n\}$.
- The encryption algorithm Enc', upon input a message $m \in \mathcal{M}$ and a public key pk, chooses $y_1,\ldots,y_n$ in $\mathbb{Z}_q$ and computes $Y_i = X_i^{y_i}$ for i in $\{1,\ldots,n\}$. It sets
$$Z_1 = g^{y_1+\cdots+y_n}, \quad Z_2 = V_1^{y_1}\cdots V_n^{y_n}, \quad k = H(Y_1,\ldots,Y_n,Z_1,Z_2)$$
and $c \leftarrow \mathsf{E}_k(m)$. The ciphertext is $(Y_1,\ldots,Y_n,c)$.
- The decryption algorithm Dec', upon input $(Y_1,\ldots,Y_n,c)$, computes
$$Z_1 = Y_1^{(x_1^{-1})}\cdots Y_n^{(x_n^{-1})}, \quad Z_2 = Y_1^{v_1}\cdots Y_n^{v_n}, \quad k = H(Y_1,\ldots,Y_n,Z_1,Z_2).$$
The plaintext is $m = \mathsf{D}_k(c) \in \mathcal{M}\cup\{\perp\}$.

Lemma 4.3.2. *Scheme Π' is correct.*

Proof. To prove correctness of the scheme Π', we have to show that, for every public key $pk = (\mathscr{G},X_1,\ldots,X_n,V_1,\ldots,V_n,H)$, where $X_i = g^{x_i}$ and $V_i = X_i^{v_i}$, and for every $y_1,\ldots,y_n$ in $\mathbb{Z}_q$, the way we compute Z_1 and Z_2 in encryption is equivalent to the way we compute them in decryption. This means that we have to prove that $Y_1^{(x_1^{-1})}\cdots Y_n^{(x_n^{-1})} = g^{y_1+\cdots+y_n}$ and $V_1^{y_1}\cdots V_n^{y_n} = Y_1^{v_1}\cdots Y_n^{v_n}$, where $Y_i = X_i^{y_i}$. The first equation is proved in Lemma 3.1.2, it remains only to prove the second part:

$$V_1^{y_1}\cdots V_n^{y_n} = X_1^{v_1y_1}\cdots X_n^{v_ny_n} = Y_1^{v_1}\cdots Y_n^{v_n}.$$

□

Theorem 4.3.3. *Suppose that (E,D) is a $(Q'_d,Q'_H,t',\varepsilon')$-CCA-secure symmetric encryption scheme, that H is modeled as a random oracle and that the strong twin n-linear DH problem is $(Q''_q,t'',\varepsilon'')$-hard relative to Ggen. Then scheme Π' from construction 3.1.1 is (Q_d,Q_H,t,ε)-secure against chosen ciphertext attacks, where $t \approx t' \approx t''$, $\varepsilon \le \varepsilon' + \varepsilon'' + \frac{Q_d}{2^{n\lambda}}$, $Q''_p \le Q_H = Q'_H$ and $Q'_d \le Q_d$.*

Proof. We will prove the theorem using the games technique. We take an adversary $\mathcal{A}$ against scheme Π running in time t and we keep it fixed in all the following games.

Game 0. The first game is the standard game for CCA-security defined in 1.2.2. Let us recall it briefly. At the beginning of the game, after having computed the public and the secret keys $(pk,sk) = \big((\mathscr{G},X_1\ldots,X_n,V_1,\ldots,V_n,H),(x_1,\ldots,x_n,v_1,\ldots,v_n)\big)$, the challenger

chooses $y_1, \ldots, y_n$ uniformly in $\mathbb{Z}_q$ and computes $(Y_1^*, \ldots, Y_n^*) = (X_1^{y_1}, \ldots, X_n^{y_n})$, $Z_1^* = g^{y_1+\cdots+y_n}$ and $Z_2^* = V_1^{y_1} \cdots V_n^{y_n}$. Then the challenger computes $k^* = H(Y_1^*, \ldots, Y_n^*, Z_1^*, Z_2^*)$. When $\mathcal{A}$ asks for the decryption of the ciphertext, the challenger uses the secret key sk to decrypt the message and gives back to $\mathcal{A}$ the corresponding plaintext. When $\mathcal{A}$ outputs two messages m_0, m_1, the challenger gives $\mathcal{A}$ the challenge ciphertext $(Y_1^*, \ldots, Y_n^*, \mathsf{E}_{k^*}(m_b))$. If $\mathcal{A}$ invokes again the decryption oracle, the challenger decrypts as above. Finally $\mathcal{A}$ outputs its guess b'. We recall that $\mathcal{A}$ has access to an oracle for the function H throughout the game.

Game 1. Game 1 works exactly like Game 0, except for the following aspect: the challenger aborts and returns $\perp$ if, before having seen the challenge ciphertext, $\mathcal{A}$ asks for the decryption of $(Y_1^*, \ldots, Y_n^*, c)$, for some ciphertext c.

Let G_i be the event that $\mathcal{A}$ wins Game i and let L be the following event: before having seen the challenge ciphertext, adversary $\mathcal{A}$ asks for the decryption of $(Y_1^*, \ldots, Y_n^*, c)$ for any c. Then, Games 1 and Game 0 are identical until L happens. By lemma 1.5.1, this means that:

$$|\Pr[G_0(\lambda) = 1] - \Pr[G_1(\lambda) = 1]| \leq \Pr[\mathsf{L}].$$

Since the values $(Y_1^*, \ldots, Y_n^*)$ are information theoretically hidden from $\mathcal{A}$'s view until it sees the challenge ciphertext, we can conclude that:

$$\Pr[\mathsf{L}] \leq \frac{Q_d}{q^n}. \tag{4.7}$$

Game 2. We define the Game 2 to be as Game 1 except for the following two aspects: when the challenger has to process the challenge ciphertext, it chooses a uniform element k in $\{0,1\}^\ell$ and outputs $(Y_1^*, \ldots, Y_n^*, \mathsf{E}_k(m_b))$. Secondly, the challenger will use this k to process every decryption query of the form $(Y_1^*, \ldots, Y_n^*, c)$, for any ciphertext c.

Consider the following event F: adversary $\mathcal{A}$ asks for

$$H(Y_1^*, \ldots, Y_n^*, \prod_{i=1}^{n} (Y_i^*)^{x_i^{-1}}, \prod_{i=1}^{n} (Y_i^*)^{v_i}).$$

Game 1 and Game 2 proceed identically until F occurs. By the difference Lemma 1.5.1, we have that:

$$|\Pr[G_2(\lambda) = 1] - \Pr[G_1(\lambda) = 1]| \leq \Pr[\mathsf{F}].$$

Claim: The probability that F happens is bounded by ε''.

To prove our claim we construct an adversary $\mathcal{B}$ against the strong twin n-linear problem as follows: when $\mathcal{B}$ is given $(\mathscr{G}, g_1, \ldots, g_n, g_1^{x_1}, \ldots, g_n^{x_n}, g_1^{y_1}, \ldots, g_n^{y_n})$ it chooses a function H and a uniform k^* in $\{0,1\}^\ell$ and it sets

$$k^* = H(g_1^{x_1}, \ldots, g_n^{x_n}, g^{x_1+\cdots+x_n}, g_1^{x_1 y_1} \cdots g_n^{x_n y_n}).$$

Then, $\mathcal{B}$ creates a table T whose entries can be of two types:

- Explicit type: $(g_1^{r_1}, \ldots, g_n^{r_n}, Z_1, Z_2, k, p) \in T$ means that $k = H(g_1^{r_1} \ldots, g_n^{r_n}, Z_1, Z_2)$ and p is the value of the predicate $n\text{-}2lin(g_1^{r_1} \ldots, g_n^{r_n}, Z_1, Z_2) \in \{0,1\}$.
- Implicit type: $(g_1^{r_1}, \ldots, g_n^{r_n}, *, *, k, 1) \in T$ means that

$$k = H(g_1^{r_1}, \ldots, g_n^{r_n}, g^{r_1+\cdots+r_n}, g_1^{r_1 y_1} \cdots g_n^{r_n y_n}).$$

Adversary $\mathcal{B}$ initializes the table T with the value $(g_1^{x_1}, \ldots, g_n^{x_n}, *, *, k^*, 1)$ and invokes $\mathcal{A}$ on input $pk = (\mathscr{G}, g_1, \ldots, g_n, g_1^{y_1}, \ldots, g_n^{y_n}, H)$.

When $\mathcal{A}$ asks for $H(g_1^{r_1}, \ldots, g_n^{r_n}, Z_1, Z_2)$, adversary $\mathcal{B}$ answers as follows:

- If there exists an entry $(g_1^{r_1}, \ldots, g_n^{r_n}, Z_1, Z_2, k, p)$ in T, $\mathcal{B}$ returns k.

- If there exists an entry $(g_1^{r_1}, \ldots, g_n^{r_n}, *, *, k, 1)$ in T and
$$n\text{-}2lin(g_1^{r_1}, \ldots, g_n^{r_n}, Z_1, Z_2) = 1$$
$\mathcal{B}$ replaces $(g_1^{r_1}, \ldots, g_n^{r_n}, *, *, k, 1)$ with $(g_1^{r_1}, \ldots, g_n^{r_n}, Z_1, Z_2, k, 1)$ in T and returns k.
- Otherwise $\mathcal{B}$ chooses k in $\{0,1\}^\ell$, returns k and stores $(g_1^{r_1}, \ldots, g_n^{r_n}, Z_1, Z_2, k, p)$ in T.

When $\mathcal{A}$ asks for the decryption of $(g_1^{r_1}, \ldots, g_n^{r_n}, c)$, $\mathcal{B}$ answers as follows:

- If there exists an entry $(g_1^{r_1}, \ldots, g_n^{r_n}, Z_1, Z_2, k, 1)$ in T, $\mathcal{B}$ returns $m = \mathsf{D}_k(c)$.
- If there exists an entry $(g_1^{r_1}, \ldots, g_n^{r_n}, *, *, k, 1)$ in T, $\mathcal{B}$ returns $m = \mathsf{D}_k(c)$.
- Otherwise $\mathcal{B}$ chooses a k uniformly in $\{0,1\}^\ell$, returns $m = \mathsf{D}_k(c)$ and stores in T
$$(g_1^{r_1}, \ldots, g_n^{r_n}, *, *, k, 1).$$

When $\mathcal{A}$ outputs two messages m_0, m_1, adversary $\mathcal{B}$ chooses a bit b in $\{0,1\}$ and gives $(g_1^{x_1}, \ldots, g_n^{x_n}, \mathsf{E}_{k^*}(m_b))$ as challenge ciphertext to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', adversary $\mathcal{B}$ ignores it and searches in T if there exists an entry of the type $(g_1^{x_1}, \ldots, g_n^{x_n}, Z_1, Z_2, k^*, 1)$. In this case $\mathcal{B}$ outputs Z_1, otherwise returns fail.

Every time event F happens, $\mathcal{B}$ solves correctly its instance of the strong twin n, linear problem, hence $\mathsf{Adv}^{Sn\text{-}2lin}_{\mathcal{B},\mathsf{Ggen}}(\lambda) \geq \Pr[\mathsf{F}]$. Also, we notice that the running time of $\mathcal{B}$ is approximately the same of the running time of $\mathcal{A}$. Since we are assuming that the strong twin n-linear DH problem is $(Q_p'', t'', \varepsilon'')$-hard, we can conclude that:

$$\Pr[\mathsf{F}] \leq \varepsilon''. \tag{4.8}$$

We also notice that adversary $\mathcal{B}$ asks oracle queries for the predicate n-$2lin$ only when adversary $\mathcal{A}$ queries the random oracle. So, $Q_p'' \leq Q_H$.

To complete the proof of the theorem we states the following:
Claim: the advantage of $\mathcal{A}$ in Game 2 is bounded by ε'.
To prove the claim consider the following adversary $\mathcal{A}'$ against the CCA-security of (E, D) that runs in about the same time of $\mathcal{A}$: upon input 1^λ adversary $\mathcal{A}'$ computes a pair of keys:

$$(pk, sk) = ((\mathscr{G}, g^{x_1}, \ldots, g^{x_n}, g^{x_1 v_1}, \ldots, g^{x_n v_n}, H), (x_1, \ldots, x_n, v_1, \ldots, v_n)) \leftarrow \mathsf{Gen}'(1^\lambda).$$

After this, adversary $\mathcal{A}'$ chooses n elements $y_1, \ldots, y_n$ in $\mathbb{Z}_q$ and computes

$$Y_i^* = g^{x_i y_i}, \quad Z_1^* = g^{y_1 + \cdots + y_n}, \quad Z_2^* = \prod_{i=1}^{n} g^{x_1 v_i y_i}.$$

Adversary $\mathcal{A}'$ sets $\hat{k} = H(Y_1^*, \ldots, Y_n^*, Z_1^*, Z_2^*)$. Next $\mathcal{A}'$ invokes $\mathcal{A}$ on input pk. When $\mathcal{A}$ asks for $H(Y_1, \ldots, Y_n, Z_1, Z_2)$, adversary $\mathcal{A}'$ simulates the random oracle as usual. When $\mathcal{A}$ asks for the decryption of $(Y_1, \ldots, Y_n, c)$, adversary $\mathcal{A}'$ answers as follows:

- If $(Y_1, \ldots, Y_n) = (Y_1^*, \ldots, Y_n^*)$ then $\mathcal{A}'$ aborts and returns $\perp$.
- Otherwise sets $Z_1 = Y_1^{x_1^{-1}} \cdots Y_n^{x_n^{-1}}$, $Z_2 = \prod_{i=1}^n Y_2^{v_i}$, computes the value $k = H(Y_1, \ldots, Y_n, Z_1, Z_2)$ and returns $m = \mathsf{D}_k(c)$.

When $\mathcal{A}$ outputs two messages m_0, m_1, adversary $\mathcal{A}'$ gives them to the challenger and receives the challenger ciphertext c^*. Then, $\mathcal{A}'$ gives $(Y_1^*, \ldots, Y_n^*, c^*)$ to $\mathcal{A}$ as challenger ciphertext. When $\mathcal{A}$ asks for more values of H, adversary $\mathcal{A}'$ processes the queries as above. When $\mathcal{A}$ asks for the decryption of $(Y_1, \ldots, Y_n, c)$, adversary $\mathcal{A}'$ answers as follows:

- If $(Y_1, \ldots, Y_n) = (Y_1^*, \ldots, Y_n^*)$ then $\mathcal{A}'$ sends c to the challenger and returns the result.

- Otherwise sets $Z = Y_1^{x_1^{-1}} \cdots Y_n^{x_n^{-1}}$, $Z_2 = \prod_{i=1}^{n} Y_2^{v_i}$, computes the value of $k = H(Y_1, \ldots, Y_n, Z_1, Z_2)$ and returns $m = \mathsf{D}_k(c)$.

Finally, $\mathcal{A}'$ outputs whatever $\mathcal{A}$ outputs.

Since $\mathcal{A}'$ breaks scheme (E, D) every time $\mathcal{A}$ wins Game 2, we have:

$$\mathsf{AdvPriv}^{\mathrm{CCA}}_{\mathcal{A}',(\mathsf{E},\mathsf{D})}(\lambda) \geq \left| \Pr[G_2(\lambda) = 1] - \frac{1}{2} \right|.$$

We also notice that the running times of $\mathcal{A}$ and $\mathcal{A}'$ are approximately the same. Since we are assuming that (E, D) is a $(Q'_d, Q'_H, t', \varepsilon')$-CCA-secure scheme, we have that

$$\left| \Pr[G_2(\lambda) = 1] - \frac{1}{2} \right| \leq \varepsilon'. \tag{4.9}$$

We notice that adversary A' queries the random oracle every time $\mathcal{A}$ does and $\mathcal{A}'$ queries the decryption oracle only if $\mathcal{A}$ sends some special ciphertext to the decryption oracle. This means that $Q'_H = Q_H$ and $Q'_d \leq Q_d$.

The theorem is proved since, putting together equations (4.7),(4.8) and (4.9), we can conclude that Π' is a $(Q_d, Q_H, t, \varepsilon)$-CCA secure scheme, where $\varepsilon \leq \varepsilon' + \varepsilon'' + \frac{Q_d}{q_n}$. □

So far, we have proved that Scheme Π' is CCA-secure under the hardness of strong twin n-linear problem. Anyway, using Corollary 4.2.4, we have that also the linear problems are sufficient for the security against chosen-ciphertext attack.

Corollary 4.3.4. *Suppose that* (E, D) *is a* $(Q'_d, Q'_H, t', \varepsilon')$*-CCA-secure symmetric encryption scheme, that* H *is modeled as a random oracle and that the* n*-linear DH problem is* (t'', ε'')*-hard relative to* Ggen*. Then* Π' *from construction 3.1.1 is* $(Q_d, Q_H, t, \varepsilon)$*-secure against chosen ciphertext attacks, where* $t \approx t' \approx t''$*,* $\varepsilon \approx \varepsilon' + \varepsilon'' + \frac{Q_d}{2^{n\lambda}}$ *and* $Q_H \leq Q'_d + Q'_H$*.*

Proof. Assume H is modeled as a random oracle and (E, D) is CCA-secure. Then, by Theorem 4.3.3, we know that Π' is secure under the hardness of the strong twin n-linear DH problem. Since by Corollary 4.2.4 we know that the strong twin n-linear ad the n-linear are equivalent, we can conclude that Π' is CCA-secure also under the hardness of the n-linear problem. □

CHAPTER 5

The Vector DH Problem

In this chapter we present a new problem, called vector DH problem. Its layout closely recalls the DH problem, but we prove that ours is strictly harder than the decisional DH problem. In particular, we show that the vector DH problem is hard in generic bilinear groups.

The vector DH problem takes inspiration from the paper [HYXZ08] by Huang et al. In that paper, a new Diffie-Hellman assumption was defined. Let us give a brief overview of their work.

Let f be a monic irreducible polynomial of degree 2 in $\mathbb{F}_q[X]$ and let A be the companion matrix corresponding to f. Then, the set $F_q[A] = \{x_0A + x_1A | x_0, x_1 \in \mathbb{F}_q\}$ is isomorphic to $\mathbb{F}_q^2$, as vector space over $\mathbb{F}_q$. They defined an action $*$ of $F_q[A]$ on $\mathbb{G}^2$, where $\mathbb{G}$ is a cyclic group of order q, and they showed that $\mathbb{G}^2$, equipped with this action, is a vector space of degree 1 over $F_q[A]$. Their new problem, called extended decisional Diffie-Hellman (EDDH) problem, is defined as follows: given a group description $\mathscr{G} = (\mathbb{G}, q, g)$, the companion matrix A of a monic irreducible polynomial in $\mathbb{F}_q[X]$ and $\mathbf{g}, X * \mathbf{g}, Y * \mathbf{g}, \mathbf{h} \in \mathbb{G}^2$ for $X, Y \in F_q[A]$, decide whether $\mathbf{h} = (XY) * \mathbf{g}$ or $\mathbf{h}$ is a uniform element in $\mathbb{G}^2$.

Our modification to the EDDH problem is as follows. We fix the non-irreducible polynomial $f(x) = x^2$ and give a simplified description of the action $*$ introducing a new product in $\mathbb{Z}_q$. In the next sections we will see that this product will allow us to define a new problem related to the Diffie-Hellman one. In this chapter, elements in $\mathbb{G}^2$ or in $\mathbb{F}_q^2$ are denoted by bold fonts. Also, we will use a double group generation Ggen that outputs $\mathscr{G} = (\mathbb{G}, q, \mathbf{g})$, where $\mathbb{G}$ as usual is a cyclic group of prime order q and $\mathbf{g} = (g_1, g_2)$, where g_1, g_2 are two generators of $\mathbb{G}$.

5.1. A new product

Given two elements $\mathbf{x} = (x_1, x_2), \mathbf{y} = (y_1, y_2) \in \mathbb{Z}_q^2$, we define a new product in $\mathbb{Z}_q^2$ in this way: $\mathbf{x}.\mathbf{y} = (x_1y_1, x_2y_1 + x_1y_2) \in \mathbb{Z}_q^2$.
With this new product, $(\mathbb{Z}_q^2, .)$ turns out to be a semigroup. In fact, the product is associative and $(1, 0)$ acts as the identity, but elements of the form $(0, x)$ are non invertible. An interesting property of this product is commutativity, in fact we notice that $\mathbf{x}.\mathbf{y}$ is symmetric in x_i and y_i; this means that $\mathbf{x}.\mathbf{y} = \mathbf{y}.\mathbf{x}$. We are pointing out this fact because it is essential that this product is commutative in order to have an encryption scheme based on the problem we are going to define in the next section. Hence, $(\mathbb{Z}_q^2, +, .)$, where $+$ is the common point-wise sum, is a commutative ring.

5.2. The Vector Diffie-Hellman Problem

We are now ready to present the new problem.
Let $\mathbb{G}$ be a cyclic group of order q. Consider $\mathbb{G}^2$ and an element $\mathbf{g} = (g_1, g_2) \in \mathbb{G}$, where g_1 and g_2 are generators of $\mathbb{G}$. Given a vector $\mathbf{x} = (x_1, x_2)$ in $\mathbb{Z}_q^2$ we define

$$\mathbf{g}^{\mathbf{x}} := (g_1^{x_1}, g_2^{x_1} g_1^{x_2}).$$

In particular, if $\mathbf{y} = (y_1, y_2)$ is another element of $\mathbb{Z}_q^2$, we have that

$$\mathbf{g}^{\mathbf{x}.\mathbf{y}} = (g_1^{x_1 y_1}, g_2^{x_1 y_1} g_1^{x_2 y_1 + x_1 y_2}).$$

PROPOSITION 5.2.1. *Let g_1, g_2 be generators of $\mathbb{G}$ and let $\boldsymbol{g} = (g_1, g_2)$. Then, the map*

$$\begin{aligned} \Psi : \mathbb{Z}_q^2 &\longrightarrow \mathbb{G}^2 \\ \boldsymbol{x} &\longmapsto \boldsymbol{g}^{\boldsymbol{x}} \end{aligned}$$

is a bijection.

PROOF. Let's prove that Ψ is surjective. Take an element $\mathbf{h} = (h_1, h_2)$ in $\mathbb{G}^2$. Then, there exist a_1, a_2 in $\mathbb{Z}_q$ such that $(h_1, h_2) = (g_1^{a_1}, g_2^{a_2})$. Now, we want an element $\mathbf{x} = (x_1, x_2) \in \mathbb{Z}_q^2$ such that $\mathbf{g}^{\mathbf{x}} = (g_1^{x_1}, g_1^{x_2} g_2^{x_1})$. Clearly we can choose $x_1 = a_1$. Then, we remain with the condition $g_1^{x_2} g_2^{x_1} = g_2^{a_2}$ which is equivalent to $g_1^{x_2} = g_2^{a_2 - a_1}$. Since g_1 is a generator of $\mathbb{G}$, there exists an $x_2 \in \mathbb{Z}_q$ such that $g_1^{x_2} = g_2^{a_2 - a_1}$. Then we have that $\mathbf{g}^{\mathbf{x}} = \mathbf{h}$. Finally, we can conclude that Ψ is bijection because $|\mathbb{Z}_q^2| = |\mathbb{G}^2|$. □

Now, using the same notation of the DH problem, we can define our new problem.
Given $\mathbf{g} = (g_1, g_2)$ in $\mathbb{G}^2$ and $\mathbf{g}^{\mathbf{x}}, \mathbf{g}^{\mathbf{y}}$, for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^2$, computing $\mathbf{g}^{\mathbf{x}.\mathbf{y}}$ is the computational vector DH problem. Anyway, we are more interested in the decisional version of this problem: given $\mathbf{g}, \mathbf{g}^{\mathbf{x}}, \mathbf{g}^{\mathbf{y}} \in \mathbb{G}^2$, for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^2$ and $h \in \mathbb{G}$, decide whether $h = \mathbf{g}^{\mathbf{x}.\mathbf{y}}[2]$ or h is a uniform element in $\mathbb{G}$ is the **decisional vector DH problem**. The reason why the decisional vector problem is defined in this unusual way is that distinguishing $\mathbf{g}^{\mathbf{x}.\mathbf{y}}$ from a uniform element in $\mathbb{G}^2$ in not harder than the decisional DH problem. Our aim is defining a problem that is hard in generic bilinear groups, differently from what happens for the DDH problem.

DEFINITION 5.2.2. The decisional vector DH problem is (t, ε)-**hard** relative to Ggen, if for every algorithm $\mathcal{A}$ whose running time is bounded by t, we have:

$$|\Pr[\mathcal{A}(\mathscr{G}, \mathbf{g}^{\mathbf{x}}, \mathbf{g}^{\mathbf{y}}, \mathbf{g}^{\mathbf{x}.\mathbf{y}}[2]) = 1] - \Pr[\mathcal{A}(\mathscr{G}, \mathbf{g}^{\mathbf{x}}, \mathbf{g}^{\mathbf{y}}, h) = 1]| \leq \varepsilon,$$

where $\mathscr{G} \leftarrow \mathsf{Ggen}(1^\lambda)$, $\mathbf{x}, \mathbf{y} \leftarrow \mathbb{Z}_q^2$ and $h \leftarrow \mathbb{G}$.

5.3. Complexity in Generic Bilinear Groups

The problem given in 5.2 may be a little hard to handle. However, there is an equivalent formulation that will allow us to give an easier proof of hardness in generic bilinear groups: given $g, h \in \mathbb{G}$ and $g^{x_1}, g^{x_2}, g^{y_1}, g^{y_2}$, for $x_i, y_i \in Z_q$, distinguish between $g^{x_1 y_2 + x_2 y_1} h^{x_1 y_1}$ and a uniform element in $\mathbb{G}$. This is completely equivalent to the decisional vector problem. In fact, let $\mathbf{g} = (g_1, g_2)$ and let $\mathbf{x} = (x_1, x_2), \mathbf{y} = (y_1, y_2)$; the element that we have to distinguish is $u = g_1^{x_1 y_2 + x_2 y_1} g_2^{x_1 y_1}$. Since g_1 is a generator of $\mathbb{G}$, there exists $\lambda \in \mathbb{Z}_q$ such that $g_1^\lambda = g_2$. Let's substitute g_1^λ in the elements above; we obtain:

$$\mathbf{g} = (g_1, g_1^\lambda), \quad \mathbf{g}^{\mathbf{x}} = (g_1^{x_1}, g_1^{x_2 + \lambda x_1}), \quad \mathbf{g}^{\mathbf{y}} = (g_1^{y_1}, g_1^{y_2 + \lambda y_1}), \quad u = g_1^{x_1 y_2 + x_2 y_1 + \lambda x_1 y_1}.$$

Now, let $x_1' = x_1,\ x_2' = x_2 + \lambda x_1,\ y_1' = y_1$ and $y_2' = y_2 + \lambda y_1$. We get:

$$\mathbf{g} = (g_1, g_1^\lambda), \quad \mathbf{g}^{\mathbf{x}} = (g_1^{x_1'}, g_1^{x_2'}), \quad \mathbf{g}^{\mathbf{y}} = (g_1^{y_1'}, g_1^{y_2'}), \quad u = g_1^{-\lambda x_1' y_1' + x_2' y_1' + x_1' y_2'}.$$

If we call $g_1 = g$ and $g_1^{-\lambda} = h$, the new formulation is the following: given $g, h^{-1}, g^{x_1'}, g^{x_2'}, g^{y_1'}$ and $g^{y_2'}$, distinguish $u = g^{x_2'y_1' + x_1'y_2'} h^{x_1'y_1'}$ from a uniform element in $\mathbb{G}$. Since giving h is equivalent to giving h^{-1}, we conclude that the new formulation is equivalent to the problem given in 5.2.

THEOREM 5.3.1. *Let $\mathbb{G}$ be a cyclic bilinear group of prime order q, let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_2$ be a bilinear map and let σ, τ be two random encodings of $\mathbb{G}, \mathbb{G}_2$, respectively. Then, for every algorithm $\mathcal{A}$ that solves the decisional vector DH problem making at most Q_g group operations queries and Q_b queries to the bilinear oracle, we have:*

$$\left| \Pr\left[\mathcal{A}\begin{pmatrix} q, \sigma(1), \sigma(z), \\ \sigma(x_1), \sigma(x_1), \\ \sigma(y_1), \sigma(y_2), \\ \sigma(t_0), \sigma(t_1) \end{pmatrix} = b \;\middle|\; \begin{array}{c} x_1, x_2, y_1, y_2, z, s \leftarrow \mathbb{Z}_q, \\ b \leftarrow \{0,1\}, t_{1-b} \leftarrow s, \\ t_b \leftarrow x_1y_2 + x_2y_1 + zx_1y_1 \end{array} \right] - \frac{1}{2} \right| \leq \frac{(Q_g + 6)^2 + Q_b^2}{q}.$$

PROOF. Let $\mathbb{G}$ be a cyclic group of order q. Let g be a generator and x_1, x_2, y_1, y_2, z be in $\mathbb{Z}_q$. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_2$ be a bilinear map.
To prove the theorem, we consider an algorithm $\mathcal{B}$ playing the following game with $\mathcal{A}$. Adversary $\mathcal{B}$ chooses 8 bit strings $\sigma_0, \ldots, \sigma_7$ uniformly in $\{0,1\}^m$. Internally, the algorithm $\mathcal{B}$ keeps track of the encoded elements using polynomials in the ring $\mathbb{Z}_q[X_1, X_2, Y_1, Y_2, Z, T_0, T_1]$. Externally, the elements that it gives to $\mathcal{A}$ are just bit strings in $\{0,1\}^m$, for a m big enough. To maintain consistency, $\mathcal{B}$ creates two lists L_1 and L_2 whose elements will be pairs (F, σ) where F is a polynomial in $\mathbb{Z}_q[X_1, X_2, Y_1, Y_2, Z, T_0, T_1]$ and σ is a bit string. The list L_1 will be for elements in $\mathbb{G}$ and the list L_2 for elements in $\mathbb{G}_2$. At the beginning, L_1 is populated by the elements

$$\{(1, \sigma_0), (X_1, \sigma_1), (X_2, \sigma_2)), (Y_1, \sigma_3), (Y_2, \sigma_4), (Z, \sigma_5), (T_0, \sigma_6), (T_1, \sigma_7)\}.$$

The list L_2 is initially empty.
The algorithm $\mathcal{B}$ starts the game providing $\mathcal{A}$ with $\sigma_0, \ldots, \sigma_7$. The simulation of the oracles goes as follows:

Group action: Given two elements σ_i, σ_j in $\mathbb{G}$, adversary $\mathcal{B}$ recovers the corresponding polynomials F_i and F_j and computes $F_i + F_j$. Then, if $F_i + F_j$ is already in L_1, adversary $\mathcal{B}$ returns to $\mathcal{A}$ the corresponding bit string, otherwise it returns a uniform element σ in $\{0,1\}^m$ and stores $(F_i + F_j, \sigma)$ in L_1. The multiplication in $\mathbb{G}_2$ is handled in the same way.

Inversion: Given en element σ in $\mathbb{G}$, adversary $\mathcal{B}$ recovers its internal representation F and computes $-F$. If the polynomial $-F$ is already in L_1, adversary $\mathcal{B}$ returns the corresponding bit string, otherwise it returns a uniform string σ' and stores $(-F, \sigma')$ in the list L_1. The inversion in $\mathbb{G}_2$ is handled analogously.

Bilinear Map: Given two elements σ_i, σ_j in $\mathbb{G}$, adversary $\mathcal{B}$ recovers the corresponding polynomials F_i and F_j and computes F_iF_j. Then, if F_iF_j is already in L_2, adversary returns to $\mathcal{A}$ the corresponding bit string, otherwise it returns a uniform element σ in $\{0,1\}^m$ and stores (F_iF_j, σ) in L_2.

Notice that all the polynomial in the first list have degree at most 1, while all the polynomials in L_2 have degree at most 2. Finally $\mathcal{A}$ will outputs a bit b'. At this point, $\mathcal{B}$ chooses uniform

values x_1, x_2, y_1, y_2, z, s in $\mathbb{Z}_q$ and a bit b, then, it sets:

$$X_1 = x_1, \quad X_2 = x_2, \quad Y_1 = y_1, \quad Y_2 = y_2, \quad Z = z, \quad T_b = x_1y_2 + x_2y_2 + zx_1x_2, \quad T_{1-b} = s.$$

We observe that the simulation provided by $\mathcal{B}$ is consistent and reveals nothing about b unless that two distinct polynomials F_1 and F_2 in L_1 or in L_2 take on the same value after the substitution.

First of all, we will show that $\mathcal{A}$ is not able to cause intentionally a collision. The values that we substitute in the variables $X_1, X_2, Y_1, Y_2, Z, T_0, T_1$ are all independent except that of T_b. This means that the only collision that $\mathcal{A}$ can engineer consists in producing $X_1Y_2 + X_2Y_2 + ZX_1X_2$ using a combination of the other polynomials in the list. Anyway, as we noticed before, the polynomials in the lists have degree at most 2, while $X_1Y_2 + X_2Y_2 + ZX_1X_2$ has degree 3. This show that $\mathcal{A}$ is not able to provoke a collision.

Now, it only remains to bound the probability that a unlucky choice of values causes a collision. We know that two polynomials F_i, F_j in L_1 have degree at most 1, so, by theorem 1.6.1, the probability that they assume the same value is at most $1/q$. For the same reason, the probability that two polynomials in L_2 take on the same value after the substitution is at most $2/q$, since every polynomial in L_2 has degree at most 2. Now, assume that $\mathcal{A}$ makes Q_g queries to the group operation oracle and Q_b queries to the bilinear oracle, then the probability ε of a collision can be bounded as follows:

$$\varepsilon \leq \binom{Q_g + 6}{2}\frac{1}{q} + \binom{Q_b}{2}\frac{2}{q} \leq \left(\binom{(Q_g + 6)}{2}\binom{Q_b}{2}\right)\frac{2}{q} \leq \frac{(Q_g + 6)^2 + Q_b^2}{q}.$$

□

5.4. A Public-key Encryption Scheme based on the Decisional Vector Problem

We can give now the construction of a scheme based on the decisional vector problem. This scheme is a generalization of the Elgamal encryption scheme and we will prove that this scheme is CPA-secure if the decisional vector problem is (t, ε)-hard.

CONSTRUCTION 5.4.1. We define a public-key encryption scheme $\Pi'' = (\mathsf{Gen}'', \mathsf{Enc}'', \mathsf{Dec}'')$ as follows:

- The key-generation algorithm Gen'', upon input 1^λ, computes a group description $\mathscr{G} = (\mathbb{G}, q, \mathbf{g}) \leftarrow \mathsf{Ggen}(1^\lambda)$, chooses a vector $\mathbf{x}$ uniformly in $\mathbb{Z}_q^2$ and returns a secret key $sk = (\mathbf{x})$ and a public key $pk = (\mathscr{G}, \mathbf{h})$, where $\mathbf{h} = \mathbf{g}^{\mathbf{x}}$. The message space is $\mathcal{M} = \mathbb{G}$.
- The probabilistic encryption algorithm Enc'', upon input $m \in \mathbb{G}$ and $pk \leftarrow \mathsf{Gen}''(1^\lambda)$, chooses a vector $\mathbf{y}$ in $\mathbb{Z}_q^2$, computes $k = \mathbf{h}^{\mathbf{y}}[2]$ and returns the ciphertext $C = (\mathbf{g}^{\mathbf{y}}, k \cdot m)$
- The deterministic decryption algorithm Dec'', on input a ciphertext $C = ((c_1, c_2), c_3)$ and a secret key $\mathbf{x} \leftarrow \mathsf{Gen}''(1^\lambda)$, computes $k' = (c_1, c_2)^{\mathbf{x}}[2]$ and returns the plaintext $m = k'^{-1} \cdot c_3$.

LEMMA 5.4.2. *The Scheme in Construction 5.4.1 is correct.*

PROOF. For correctness, we have to show that, for every pairs of keys

$$(sk, pk) = \big((x_1, x_2), \big((\mathbb{G}, q, (g_1, g_2)), (h_1, h_2)\big)\big) \leftarrow \mathsf{Gen}''(1^\lambda)$$

and for every (y_1, y_2) in $\mathbb{Z}_q^2$, the key k computed in encryption is equal to the key k' computed in decryption. In encryption k is calculated as follows:

$$k = \mathbf{h}^{\mathbf{y}}[2] = h_1^{y_2}h_2^{y_1} = (g_1^{x_1})^{y_2}(g_1^{x_2}g_2^{x_1})^{y_1} = g_1^{x_1y_2 + x_2y_1}g_2^{x_1y_1}.$$

Now, let $(g_1, g_2)^{y_1, y_2} = (c_1, c_2)$, we have that:

$$k' = (c_1, c_2)^{\mathbf{x}}[2] = c_1^{x_2} c_2^{x_1} = (g_1^{y_1})^{x_2} (g_1^{y_2} g_2^{x_1})^{x_1} = g_1^{x_1 y_2 + x_2 y_1} g_2^{x_1 y_1}.$$

This show that $k = k'$ and, in particular, that scheme Π'' from Construction 5.4.1 is correct. □

THEOREM 5.4.3. *If the decisional vector problem is (t, ε)-hard, then scheme in Construction 5.4.1 is (t, ε)-secure against chosen-plaintext attack.*

PROOF. Let $\mathcal{A}$ be an adversary against the scheme that runs in time t.

Game 0. Let Game 0 be the standard game for CPA-security defined in 1.2.1. The public key is $pk = (\mathscr{G}, \mathbf{g}^{\mathbf{x}})$ and for the challenge ciphertext, the challenger chooses $\mathbf{y} \leftarrow \mathbb{Z}_q^2$ and sets $C^* = (\mathbf{g}^{\mathbf{y}}, k \cdot m_b)$, where $k = \mathbf{g}^{\mathbf{x} \cdot \mathbf{y}}[2]$.

Game 1. Game 1 works exactly like Game 0 except in the following aspect: while computing the challenge ciphertext, the challenger chooses also an element $h \leftarrow \mathbb{G}$ and returns the ciphertext $C^* = (\mathbf{g}^{\mathbf{y}}, h \cdot m_b)$.

Consider the following adversary $\mathcal{B}$ against the decisional vector problem: upon input $(\mathscr{G}, \mathbf{g}^{\mathbf{x}}, \mathbf{g}^{\mathbf{y}}, h)$, adversary $\mathcal{B}$ invokes $\mathcal{A}$ on input $pk = (\mathscr{G}, \mathbf{g}^{\mathbf{x}})$. When $\mathcal{A}$ outputs two messages m_0, m_1, adversary $\mathcal{B}$ chooses a bit b in $\{0, 1\}$ and sends the challenge ciphertext $C^* = (\mathbf{g}^{\mathbf{y}}, h \cdot m_b)$ to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', adversary $\mathcal{B}$ outputs 1 if and only if $b' = b$.

Let us analyze the behavior of $\mathcal{B}$:

Case 1: The input of $\mathcal{B}$ is $(\mathbf{g}, \mathbf{g}^{\mathbf{x}}, \mathbf{g}^{\mathbf{y}}, \mathbf{g}^{\mathbf{x} \cdot \mathbf{y}}[2])$. In this case, adversary $\mathcal{A}$ is essentially playing Game 1. Then:

$$\Pr[\mathcal{B}((\mathbf{g}, \mathbf{g}^{\mathbf{x}}, \mathbf{g}^{\mathbf{y}}, h)) = 1] = \Pr[G_0(\lambda) = 1]. \tag{5.1}$$

Case 2: The input of $\mathcal{B}$ is $(\mathbf{g}, \mathbf{g}^{\mathbf{x}}, \mathbf{g}^{\mathbf{y}}, h')$, where h' is a uniform element in $\mathbb{G}$. In this case, the view of $\mathcal{A}$ is identically distributed to $\mathcal{A}$'s view in Game 1. Moreover, since h' is chosen randomly in $\mathbb{G}$, also $m_b \cdot h$ will be a uniform element in $\mathbb{G}$, hence the challenge ciphertext received by $\mathcal{A}$ contains no information about the bit b chosen by the challenger. This means that:

$$\Pr[\mathcal{B}((\mathbf{g}, \mathbf{g}^{\mathbf{x}}, \mathbf{g}^{\mathbf{y}}, h')) = 1] = \Pr[G_1(\lambda) = 1] = \frac{1}{2}. \tag{5.2}$$

Equations (5.1) and (5.2) tell us that:

$$\begin{aligned} &|\Pr[\mathcal{B}((\mathbf{g}, \mathbf{g}^{\mathbf{x}}, \mathbf{g}^{\mathbf{y}}, h)) = 1] - \Pr[\mathcal{B}((\mathbf{g}, \mathbf{g}^{\mathbf{x}}, \mathbf{g}^{\mathbf{y}}, h')) = 1]| \\ =&|\Pr[G_0(\lambda) = 1] - \frac{1}{2}| = \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}, \mathrm{vec}}(\lambda) \end{aligned}$$

Since we are supposing that the decisional vector problem is (t, ε)-hard and since the running time of $\mathcal{B}$ is approximately the same of $\mathcal{A}$'s running time, we can conclude that $\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{A}, \Pi''}(\lambda) \leq \varepsilon$. This proves that scheme Π'' is (t, ε)-CPA secure. □

Bibliography

[ABR01] M. Abdalla, M. Bellare, P. Rogaway. DHIES: An encryption scheme based on the Diffie-Hellman Problem. In *Topics in Cryptology* volume 2020, of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[BBG05] D. Boneh, X. Boyen, E. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptography-Eurocrypt'05*, volume 3493 of *Lecture Notes in Computer Science*. Springer, 2005.

[BBS04] D. Boneh, X. Boyen, H. Shacham. Short group signatures. In *Proceedings of Crypto 2004*, volume 3152 of LNCS. Springer-Verlag, 2004.

[BDJR98] M. Bellare, A. Desai, D. Pointcheval, P. Rogaway. Raletions among notions of security for public-key encryption schemes. In *Advances in Cryptography-Crypto'98*, volume 1462 of *Lecture Notes in Computer Science*. Springer,1998.

[BKR00] M. Bellare, J. Kilian, P. Rogaway. The security of the cipherblock chaining message authentication code. *Journal of Computer and System Sciences,* 2000.

[Ble98] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS#1. In *Advances in Cryptography-Crypto'98*, volume 1462 of *Lecture Notes in Computer Science*. Springer,1998.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *1st ACM Conf. on Computer and Communication Security.* ACM, 1993.

[BR09] M. Bellare and P. Rogaway. Simulation without the artificial abort: simplified proof and improved concrete security for Waters' IBE scheme. In *Cryptology ePrint Archive*, Report 2009/084, 2009.

[Ca09] D. Cash. On the security and on the efficiency of encryption. PhD thesis, Georgia Institute of Technology, 2009.

[CGH04] R. Canetti, O. Goldreich, S. Halevi. The random oracle methology, revisited. In *Journal of the ACM*, 2004.

[CKS08] D. Cash, E. Kiltz, V. Shoup. The twin Diffie-Hellman problem and applications. In *Cryptology ePrint Archive*, Report 2008/067, 2008.

[CS98] R. Cramer and V. Shoup. A practical Public Key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptography-CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*. Springer, 1998.

[CS03] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure under adaptive chosen ciphertext attack. *SIAM Journal on Computing,* 2003.

[De02] A. W. Dent.Adapting the weaknesses of the random oracle model to the generic group model. In *Advances in Cryptology – Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, Springer-Verlag, 2002.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Information Theory,* 1976.

[GM84] S. Goldwasser and S. Micali. Probabilistic Encryption. In *Journal of Computer and System Sciences*, 1984.

[HK07] D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In *Advances in Cryptography-CRYPTO '07*, volume 4622 of *Lecture Notes in Computer Science*. Springer, 2007.

[HYXZ08] H. Huang, B. Yang, S. Zhu, G. Xiao. Generalized Elgamal public key cryptosystem based on a new Diffie-Hellman problem. Springer-Verlag Berlin, 2008.

[JN01] A. Joux and K. Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. In *Cryptology ePrint Archive*, Report 2001/003, 2001.

[KL08] J. Katz and Y. Lindell. Introduction to modern cryptography. Chapman & Hall/CRC, 2008.

[NY90] M. Naor and M. Yung. Public-key Cryptosystems provably secure against chosen ciphertext attacks. In *Proc. 21st Annual ACM Symposium on Theory of Computing.* ACM, 1990.

[PH78] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Information Theory,* 1978.

[RS92] C. Rackoff, D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, Springer, 1992.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. In *Journal of ACM.* ACM, 1980

[Sha07] H. Shacham. A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. In *Cryptology ePrint Archive*, Report 2007/074, 2007.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptography-Eurocrypt '97*, volume 1233 of *Lecture Notes in Computer Science*. Springer, 1997.

[Sho98] V. Shoup. Why chosen ciphertext security matters. Technical report RZ 3076, IBM Zurich, 1998. Available at `http://shoup.net/papers/expo.pdf`.

[Sho01] V. Shoup. A proposal for an ISO standard for public key encryption. 2001. Available at `http://shoup.net/papers/iso-2_1.pdf`.

[Sho04] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. 2004.