On Diffie-Hellman–like Security Assumptions

Antoine Joux[1] and Antoine Rojat[2]

CryptoExperts[1] and
Laboratoire PRISM - Université de Versailles Saint-Quentin-en-Yvelines[1,2]
45 avenue des États-Unis, F-78035 Versailles Cedex, France
antoine.joux@m4x.org[1] , aro@prism.uvsq.fr[2]

Abstract. Over the past decade bilinear maps have been used to build a large variety of cryptosystems. In parallel to new functionalities, we have also seen the emergence of many security assumptions. This leads to the general question of comparing two such assumptions. Boneh, Boyen and Goh introduced the *Uber assumption* as an attempt to offer a general framework for security assessment. Their idea is to propose a generic security assumption that can be specialized to suit the needs of any proof of protocols involving bilinear pairing. Even though the *Uber assumption* has been only stated in the bilinear setting, it can be easily restated to deal with ordinary Diffie-Hellman groups and assess other type of protocols.
In this article, we explore some particular cases of the Uber assumption; namely the n-CDH-assumption, the n^{th}-CDH-assumption and the Q-CDH-assumption. We analyse the relationships between those cases and more precisely from a security point of view. Our analysis does not rely on any special property of the considered group(s) and does not use the generic group model.

Keywords: Diffie-Hellman, Computational, Reduction, Groups, Pairing, Adversarial Oracle

1 Introduction

There are many different ways of analyzing the security of a cryptographic scheme. One can use a formal proof system (as in [8]) or perform a direct more concrete and computationally oriented proof or use a game based kind of proof (see [6] or [5] for the case of computer-assisted proofs). A difficulty with formal proofs is that the security hypothesis often needs to be stated in an abstract black-box way which can be difficult to verify on a concrete scheme or implementation. For this reason, most proofs of cryptographic schemes emphasize the computational aspect; this is, in particular, the case of reductionist proofs.
In general, a cryptosystem is said to have reductionist or computational security when its security requirements can be stated in an adversarial model with clear assumptions about the adversary, its means of manipulating the system and its computational resources. In this approach, the security of a cryptographic scheme is based on some core algorithmic problem which is assumed to be hard to solve. The scheme should remain secure as long as the chosen instances of the underlying algorithmic problem remain hard.
Among the classical security assumptions used in public-key cryptography, we find the discrete logarithm problem proven difficult in the generic group model by Shoup [24] or the Diffie-Hellman assumption which underlies Diffie-Hellman key exchange protocol [14]. There is a wide variety of more specialized assumptions which have been introduced over the years. In this article, we focus on assumptions related to Diffie-Hellman: they are defined in section 2.
These Diffie-Hellman related assumptions can either involve a single group or can be stated in a richer bilinear (or pairing-based) setting. Initially, bilinear pairings were used for cryptanalytic purposes for example the MOV attack [22]. Basically, the attacks using bilinear pairings reduce the discrete logarithm problem on an elliptic (or hyperelliptic) curve to the discrete logarithm problem in a finite field.
More recently, bilinear pairings have been used to construct new cryptographic primitives. In [19], Joux showed that the bilinear pairings can be used constructively, proposing to use them to construct a tripartite one-round Diffie-Hellman key agreement protocol. At Crypto 2001, Boneh and Franklin [10] used pairings to propose the first fully functional, efficient and provably secure identity-based encryption scheme. At Asiacrypt 2001, Boneh, Lynn and Shacham [11] proposed a pairing-based signature scheme that features the shortest length among known signature schemes.

One sometime confusing aspect of pairing-based cryptography is that a large number of ad'hoc security assumptions have been introduced in parallel with the new schemes and protocols. In particular, it is not easy to compare the different security assumptions and, thus, to compare the security level of schemes based on different assumptions. As an attempt to simplify the situation, Boneh, Boyen and Goh have introduced the *Uber assumption* in [9]. This assumption offers a general framework which can host all previous assumptions. This assumption is stated as follows:

Definition 1 (Uber assumption ([1] section 5)).
Let p be some large prime, and let r, s, t, and c be four positive integers. Consider $R \in \mathbb{F}_p[X_1, \cdots, X_c]^r$, $S \in \mathbb{F}_p[X_1, \cdots, X_c]^s$, and $T \in \mathbb{F}_p[X_1, \cdots, X_c]^t$, three tuples of multivariate polynomials over the field $\mathbb{F}_p$, and respectively containing r, s, and t polynomials in the same c variables $X_1, \cdots, X_c$. We write $R = \langle r_1, r_2, \cdots, r_r \rangle$, $S = \langle s_1, s_2, \cdots, s_s \rangle$ and $T = \langle t_1, t_2, \cdots, t_t \rangle$. The first components of R, S, and T are forced to the constant polynomial 1; that is, $r_1 = s_1 = t_1 = 1$. For a set Ω, a function $f : \mathbb{F}_p \to \Omega$, and a vector $\langle x_1, \cdots, x_c \rangle \in \mathbb{F}_p^d$, we use the notation $f(R)$ to denote the application of f to each element of R, namely, $f(R(x_1, \cdots, x_c)) = \langle f(r_1(x_1, \cdots, x_c)), \cdots, f(r_r(x_1, \cdots, x_c)) \rangle \in \Omega^r$; and use a similar notation for applying f to the s-tuple S and the t-tuple T. Let then $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{H}$ be cyclic groups of order p, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{H}$ be a cryptographic bilinear pairing. Suppose that $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ respectively generate the groups to which they belong, and set $h = e(g_1, g_2) \in \mathbb{H}$ thus generating $\mathbb{H}$. Together, these form the bilinear context $G = \langle p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{H}, g_1, g_2, e \rangle$. The (R,S,T,f)-Diffie-Hellman problem in G is defined as follows. Given the input vector,

$$U(x_1, \cdots, x_c) = \left(g_1^{R(x_1, \cdots, x_c)}, g_2^{S(x_1, \cdots, x_c)}, h^{T(x_1, \cdots, x_c)} \right) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{H},$$

secretly created from random $x_1, \cdots, x_c \in \mathbb{F}_p$, compute the output value,

$$h^{f(x_1, \cdots, x_c)} \in \mathbb{H}.$$

In order to state the difficulty of the *Uber assumption*, Boyen, Boneh and Goh have introduced a notion of dependency as follows:

Definition 2 (dependence ([1] section 5)).
Let $R = \langle r_1, \cdots r_n \} \in \mathbb{F}_p[X_1, \cdots, X_c]^r, S = \langle s_1, \cdots s_n \} \in \mathbb{F}_p[X_1, \cdots, X_c]^s$, and $T = \langle t_1, \cdots t_n \} \in \mathbb{F}_p[X_1, \cdots, X_c]^t$ as defined in definition 1.
We say that a polynomial $f \in \mathbb{F}_p[X_1, \cdots, X_c]$ is dependent on the triple $\langle R, S, T \rangle$ if there exists $rs + t$ constants $\{a_{i,j}\}_{1 \le i \le r 1 \le j \le s}$ and $\{b_k\}_{1 \le k \le t}$ and possibly $r^2 + s^2$ $\left\{d_{n,m}^1\right\}_{1 \le n,m \le r}$ and $\left\{d_{p,q}^{1,2}\right\}_{1 \le p,q \le s}$ additional constants such that:

$$f = \sum_{i=1}^{r} \sum_{j=1}^{s} a_{i,j} r_i s_j + \sum_{n=1}^{r} \sum_{m=1}^{r} d_{n,m}^1 r_n r_m + \sum_{p=1}^{s} \sum_{q=1}^{s} d_{p,q}^{1,2} s_p s_q + \sum_{k=1}^{t} b_k t_k$$

When a polynomial f is not dependent from a triple $\langle R, S, T \rangle$, we say that this polynomial is indenpendent from this triple.

Recalling what is stated in the paper of Boyen [1], the constants $d_{p,q}^{1,2}$ models the existence of the computable isomorphism $\varphi : \mathbb{G}_1 \to \mathbb{G}_2$ which exists for pairings of **Type 1** and **Type 2**. While the constants $d_{n,m}^1$ models the existence of the inverse isomorphism $\varphi^{-1} : \mathbb{G}_2 \to \mathbb{G}_1$ which exists for pairings of type 1. When neither φ^{-1} nor φ exists, Boyen simply proposed to set the $d_{n,m}^1$ and the $d_{p,q}^{1,2}$ constants to zero which models the pairings of **Type 3**. In there paper, they stated that the *Uber assumption* holds whenever the polynomial f is not dependent from the triple $\{R, S, T\}$.. The generic form of the *Uber assumption* introduces a security assumption that allows virtually any of the previous assumptions to be reformulated as a sub-case of it. As a consequence, it suffices to rely on this generic assumption and it is no longer necessary to introduce additional assumptions. Moreover, the *Uber assumption* has been proven secure in the generic group model (see the appendix of [9]). However, since the *Uber assumption* subsumed previous assumption, it is very strong and it might feel riskier to rely on it than on a simpler assumption.

Our contribution. In this article, we consider several assumptions that are specific sub-cases of the *Uber assumption*. More precisely, we focus on hypothesis where the sets R, S and T are:

$$S = \{X_1, \cdots, X_n\} \text{ and } R = \{X_n + 1, \cdots, X_s\} \text{ and } T = \emptyset.$$

Our goal is to understand more precisely the relations between the different security assumptions that can be derived from this kind of sets.

First, we analyze this kind of assumptions in the group setting, that is $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}_3$ and there is no bilinear pairing. We begin the article with that kind of assumption because the proofs are easier to follow when we consider the group setting. All the proofs can be adapted quite easily to the bilinear setting thus we analyze the security relationships in the bilinear setting without recalling the proofs.

We distinguish three kinds of hypothesis depending on the form of the polynomial f. Firs we consider the case where $f = X_1 X_2 \cdots X_s$ then, we focus on the case where $f = X_1^s$ and finally, we consider the case where f is any arbitrary polynomial of degree less or equal than s.

Throughout the different proofs, we wish to compare the difficulty of an hypothesis H_1 and another hypothesis H_2. More precisely, we want to prove relationships of the form $H_1 \Leftarrow H_2$, i.e. to prove that H_1 is actually harder than H_2. In order to do so, we first assume that we have access to an oracle $\mathscr{O}_{H_2}$ that can solve the H_2 hypothesis for any instance of H_2, $\mathscr{O}_{H_2}$ proposes a solution, which should be correct with probability at least $1 - \varepsilon$. Using this oracle we construct an algorithm that solves H_1.
Along this paper, we will use three kind of oracles:

Definition 3 (Perfect oracle).
A perfect oracle is an oracle that, when queried, always answers the correct solution.

We can consider another class of oracle that is somehow randomize:

Definition 4 (Almost perfect oracle).
An almost perfect oracle is an oracle that, when queried, answers the correct solution with a probability at least $1 - \varepsilon$ such that ε is exponentially small that is:

$$\exists c \in \mathbb{R}_+^\star \ : \ \varepsilon < \exp(-c).$$

Finally, one can remark that a general difficulty is that the an oracle can behave arbitrarily on the ε fraction of incorrect answers. In particular, it may behave in a malicious way designed to adversarily affect our algorithm. This kind of oracle can be formally defined as follows:

Definition 5 (Adversarial oracle).
An adversarial oracle is an oracle that, when queried, answers correctly with a probability at least $1 - \varepsilon$. When the oracle does not answers correctly it can adversarily adapt its answers in a malicious way.

Normally, with discrete logarithm based assumptions, the possibly malicious behavior of adversarial oracles can be ignored. This is due to the classical property of random self-reducibility see [2] and [15]. One of our important contribution is to propose a new form of random self-reducibility versatile enough to deal with the different assumptions we are considering (see Theorem 3). This allows us to reduce the probability of error to $O(1/p)$ were p is the cardinality of the considered groups. This allows us to complete the work done by Bao, Deng and Zhu in [3] where they consider some variations of the Diffie-Hellman problems. We extend their work by considering other kind of problems and furthermore, we do not rely on perfect oracle (see definition 3 below) as they did in their paper.
In the group setting, we are able to prove that all the assumptions we are considering are in fact equivalent — in terms of hardness — to the standard computational Diffie-Hellman assumption. Our results are summarized on Figure 1.

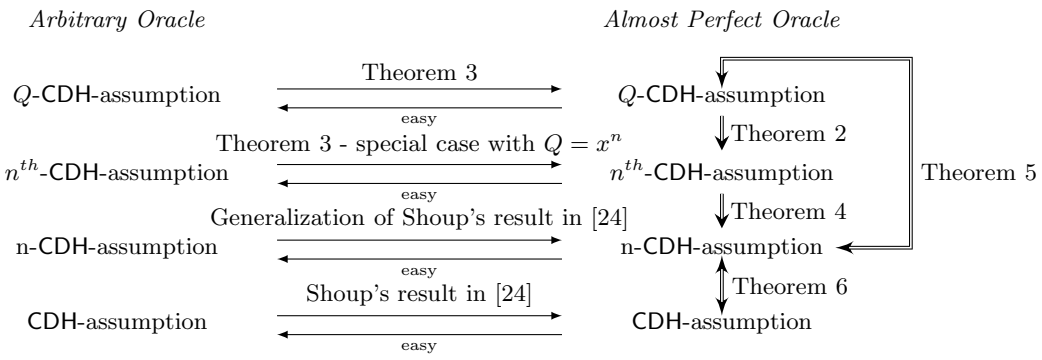

Fig. 1: Security assumptions relations (Group setting)

We obtain a very different situation in the bilinear setting: a hierarchy appears between the different assumptions (see figures 3 and 4). It is based on the degree of the polynomial f. The higher the degree of f is, the harder the assumption. In particular, the case where $f = X_1 \cdots X_S$ cannot be reduced to the classical CBDH-assumption. The paper is organized as follows. In Section 2, we introduce the necessary notations and definitions. The different sub cases of the *Uber assumption* we are considering are redefined as ad'hoc security assumptions in section 3 In section 4, we prove that a general oracle that can predict the value in the exponent of a polynomial Q can be used to solve powers (up to the degree d- in the exponent). In Section 5, we show how to compute products in the exponents using an oracle for powers. Section 6 closes the loop, moving from products to arbitrary polynomials. Finally, in section 7, we explain how the previous results adapt to the bilinear setting.

2 Notations

Group notations
The different assumptions we analyze are defined over groups of prime order. This restriction allows us to focus on the main difficulty of the reduction proofs. Most properties can be generalized to composite group order, using either Chinese remainder or Hensel lifting arguments.
As usual, the security of the different assumptions are strongly related to the group size; if the group size is too small then exhaustive search or birthday based attacks become feasible. Our assumptions do not require any specific property from the considered groups. As a consequence, we simply assume that we are given a black-box description of a prime order group as $\mathbb{G}$ with generator g.

Bilinear group notations
Before we set the notations for the bilinear groups, let us recall the definition of a cryptographic bilinear pairing.

Definition 6.
A cryptographic bilinear pairing is a function denoted $\hat{e}$ from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{H}$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{H}$ are groups with the same order. The function $\hat{e}$ verifies the following properties:

Non degeneracy *Let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, then $h = \hat{e}(g_1, g_2)$ is a generator of $\mathbb{H}$.*
Bilinearity $\forall x, y \in \mathbb{Z}$, $\hat{e}(g_1^x, g_2^y) = h^{xy}$.
Efficiently computable *There exists an efficient algorithm that on input $(u, v) \in \mathbb{G}_1 \times \mathbb{G}_2$ computes $\hat{e}(u, v)$.*

In practice, cryptographic bilinear pairing instances are obtained using the Weil, Tate or a related pairing on algebraic curves over finite fields (see [7,16,4,13]). All these pairings can be efficiently computed thanks to Miller's algorithm [23].
In [17], Galbraith, Paterson and Smart introduced the idea of classifying the bilinear pairing according to the hardness of computing an isomorphism $\Psi : \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ and its inverse $\Psi^{-1} : \mathbb{G}_2 \longrightarrow \mathbb{G}_1$. According to this classification, we say that $\hat{e}$ is of:

Type 1 If both Ψ and Ψ^{-1} are efficiently computable (this includes the case where both groups are equal);
Type 2 If Ψ is efficiently computable, but not Ψ^{-1} (if the converse holds, we just swap the notation);
Type 3 If neither Ψ nor Ψ^{-1} is efficiently computable.

For simplicity of exposure, we formalize our definitions and theorems in the *Type 1* setting. This allows us to focus on the general ideas of the reduction proofs and omit the extra technical details that appear when dealing with other type of pairings. As in the classical group context, the definitions of our assumptions do not require any specific property of a bilinear friendly group. Thus, we consider that $\mathbb{G}$ and $\mathbb{H}$ are two black-box groups of prime order, with respective generators g and h. We further assume that $\hat{e}$ is a black-box bilinear pairing such that $h = \hat{e}(g, g)$.

3 Security assumptions

As stated in the introduction, we focus on security relationships among several sub-cases of the *Uber assumption*. In this section we define the different assumptions we analyzed as stand alone assumptions. All the assumptions we consider follow the same definition pattern. Before stating this definition pattern, we recall the notion of a negligible function:

Definition 7 (Negligible function).
A function $f : \mathbb{N} \to [0, 1]$ is called negligible if for every positive polynomial p, there exists some $k_0 \in \mathbb{N}$ such that for every $k \geq k_0$, $|f(k)| < 1/p(k)$.

We now present the definition pattern we use throughout this paper:

Definition Pattern.
Let $\mathcal{P}$ denote the name of the assumption we are defining. We say that $(\eta, \nu) - \mathcal{P}$ holds if for all probabilistic polynomial time adversaries, $\mathscr{A}_\eta$ running in time η, the probability ν that, when queried on any random input $\mathcal{I}$, the algorithm output the correct solution for $\mathcal{P}$ denoted $\mathcal{P}(\mathcal{I})$ is negligible:

$$\forall \mathscr{A}_\eta : \mathsf{Pr}\left[\mathscr{A}_\eta(\mathcal{I}) = \mathcal{P}(\mathcal{I})\right] \textit{ is negligible.}$$

When we sample independently random values $x_1, \cdots, x_n$ in a set S we write $x_1, \cdots, x_n \in_R S$.
We state the different assumptions we are considering below.

Definition 8 (η-CDH-assumption).

$$\forall \mathscr{A}_\eta, \forall x, y \in_R \mathbb{F}_p : \mathsf{Pr}\left[\mathscr{A}_\eta(g^x, g^y) = g^{xy}\right] \textit{ is negligible.}$$

This assumption has first been introduced by Diffie and Hellman in [14]. It was the first realisation of a cryptographic protocol allowing secure communication between two parties without requiring prior knowledge. In the bilinear context, this problem has been first introduced by Joux in [19] and is defined as follows:

Definition 9 (η-CBDH-assumption).

$$\forall \mathscr{A}_\eta, \forall x, y, z \in_R \mathbb{F}_p : \mathsf{Pr}\left[\mathscr{A}_\eta(g^x, g^y, g^z) = h^{xyz}\right] \textit{ is negligible.}$$

The CBDH-assumption have been widely studied and several cryptographic protocols have been proven secure by reduction to this hypothesis such as the first identity based protocol introduced by Boneh and Franklin in [10]. We consider another problem that is a generalisation of the classical CDH-assumption the n-CDH-assumption in which instead of being given only two quantities, we are given n random elements of a group of the form g^{x_i} and we have to compute $g^{\prod x_i}$. More formally we define then-CDH-assumption as follows:

Definition 10 (η-n-CDH-assumption).

$$\forall \mathscr{A}_\eta, \forall x_1, \cdots, x_n \in_R \mathbb{F}_p : \mathsf{Pr}\left[\mathscr{A}_\eta(g^{x_1}, \cdots, g^{x_n}) = g^{x_1 \cdots x_n}\right] \textit{ is negligible.}$$

To the best of our knowledge, this assumption has not already been used in order to build a cryptographic protocol. The corresponding bilinear assumption is defined as follows:

Definition 11 (η-n-CBDH-assumption).

$$\forall \mathscr{A}_\eta, \forall x_1, \cdots, x_n \in_R \mathbb{F}_p : \mathsf{Pr}\left[\mathscr{A}_\eta(g^{x_1}, \cdots, g^{x_n}) = h^{x_1 \cdots x_n}\right] \text{ is negligible.}$$

In addition to the two extensions of the Computational Diffie-Hellman extension, we have also considered the case of power exponents problem, i.e, given a group element of the form g^x one has to compute g^{x^n}. This problem have been used in [21] and in [12] with $n = 2$ and is called *Square exponent problem* more formally we define the n^{th}-CDH-assumption as follows:

Definition 12 (η-n^{th}-CDH-assumption).

$$\forall \mathscr{A}_\eta, \forall x \in_R \mathbb{F}_p : \mathsf{Pr}\left[\mathscr{A}_\eta(g^x) = g^{x^n}\right] \text{ is negligible.}$$

The corresponding bilinear assumption can be stated as follows:

Definition 13 (η-n^{th}-CBDH-assumption).

$$\forall \mathscr{A}_\eta, \forall x \in_R \mathbb{F}_p : \mathsf{Pr}\left[\mathscr{A}_{(\nu,\eta)}(g^x) = h^{x^n}\right] \text{ is negligible.}$$

Finally we have introduce a new assumption the Q-CDH-assumption in which given some inputs of the form g^{x_i} and a polynomial Q, one has to compute $g^{Q(x_1,\cdots,x_n)}$. More formally we define the Q-CDH-assumption as follows:

Definition 14 (η-Q-CDH-assumption).
Let Q be a polynomial of $\mathbb{F}_p[X_1, \cdots, X_n]$ of degree d.

$$\forall \mathscr{A}_\eta, \forall x_1, \cdots, x_n \in_R \mathbb{F}_p : \mathsf{Pr}\left[\mathscr{A}_\eta(g^{x_1}, \cdots, g^{x_n}) = g^{Q(x_1,\cdots,x_n)}\right] \text{ is negligible.}$$

The corresponding bilinear assumption can be defined as follows:

Definition 15 (η-Q-CBDH-assumption).
Let Q be a polynomial of $\mathbb{F}_q[X_1, \cdots, X_n]$ of degree d.

$$\forall \mathscr{A}_\eta, \forall x_1, \cdots, x_n \in_R \mathbb{F}_p : \mathsf{Pr}\left[\mathscr{A}_\eta(g^{x_1}, \cdots, g^{x_n}) = h^{Q(x_1,\cdots,x_n)}\right] \text{ is negligible.}$$

4 From polynomials to powers

In this section, we focus on the main difficulty of the article: the reduction from the Q-CDH-assumption to the n^{th}-CDH-assumption. We conduct our analysis in the group setting since it is easier to follow and the notations are also clearer. The section is organized as follows. First we consider that we are given access to a perfect oracle that solves the Q-CDH-assumption. And we use it to build an algorithm that solves the n^{th}-CDH-assumption. And then, we explain how it is possible to build a random oracle solving the Q-CDH-assumption when having access only to an adversarial oracle that solves the Q-CDH-assumption.

4.1 Using a perfect Oracle

In this subsection, our goal is to compute g^{x^n} for any $n \leq degree(Q) = d$ knowing g^x and having access to a perfect oracle that on input $g^{x_1}, \cdots, g^{x_n}$ outputs $g^{Q(x_1,\cdots,x_n)}$. Our result is summarized in theorem 2:

Theorem 1. *Given access to an oracle that solves the Q-CDH-assumption, we can, after performing $d + 1$ oracle calls, solve all of the n^{th}-CDH-assumption with $n \leq d$, where d is the degree of polynomial Q.*

Proof. First, we independently sample n random values $\lambda_1^1, \cdots, \lambda_n^1$ and we call the perfect oracle on the following instance:

$$\left(g^{x+\lambda_1^1}, \cdots, g^{x+\lambda_n^1}\right).$$

This sampling will prove to be essential when dealing with an adversarial oracle. It is not strictly necessary with a perfect oracle, but for the sake of clarity, it is easier to keep the same overall strategy in both cases.
The response of the oracle to this randomized query is, by definition of the oracle,

$$g^{Q(x+\lambda_1^1, \cdots, x+\lambda_n^1)}.$$

As the polynomial Q is formally known, the polynomial $Q(x+\lambda_1^1, \cdots, x+\lambda_n^1)$ can be rewritten as polynomial in x:

$$Q(x+\lambda_1^1, \cdots, x+\lambda_n^1) = \sum_{j=0}^{d} [\mathsf{Q}(\lambda_1^1 + x, \cdots, \lambda_n^1 + x)]_j \cdot x^j.$$

Where $[\mathsf{Q}(\lambda_1^i + x, \cdots, \lambda_n^i + x)]_j$ denotes the coefficient of x^j in the polynomial $Q(x+\lambda_1^i, \cdots, x+\lambda_n^i)$.
Note that the terms of degree 0 and degree 1 could be removed since they can be computed from the knowledge of the λ_i and g^x. But in order to be able to prove our argument, they are required in the matrix.
By sampling enough values, we can build the following matrix:

$$\mathcal{M} = \begin{bmatrix} [\mathsf{Q}(\lambda_1^1 + x, \cdots, \lambda_n^1 + x)]_0 & \cdots & [\mathsf{Q}(\lambda_1^1 + x, \cdots, \lambda_n^1 + x)]_d \\ [\mathsf{Q}(\lambda_1^2 + x, \cdots, \lambda_n^2 + x)]_0 & \cdots & [\mathsf{Q}(\lambda_1^2 + x, \cdots, \lambda_n^2 + x)]_d \\ \vdots & \vdots & \vdots \\ [\mathsf{Q}(\lambda_1^{d+1} + x, \cdots, \lambda_n^{d+1} + x)]_0 & \cdots & [\mathsf{Q}(\lambda_1^{d+1} + x, \cdots, \lambda_n^{d+1} + x)]_d \end{bmatrix}$$

This matrix is computed without the need to call any oracle, it is derived formally from the known coefficients λ_i^j. In order to be able to prove our result, we need this matrix to be non-singular. We use the following technical lemma:

Lemma 1. *The matrix $\mathcal{M}$ is non-singular with probability at least $1 - 2/p$.*

The proof of this lemma is given in appendix (see appendix A). With reasonnable parameters choices, this probability is non-negligible. Thus, as the λ_i^j are randomly sampled, $\mathcal{M}$ can be assumed to be non singular.
Now, recall that this matrix is formally computed from the randomly sampled values and the knowledge of the polynomial Q. Each time we compute one row of this matrix, we query the oracle in parallel. When computing the i^{th} row, we query the oracle on the following input:

$$\left(g^{x+\lambda_1^i}, \cdots, g^{x+\lambda_n^i}\right).$$

Let $\tilde{z}_i$ be the output of the oracle. We store all those values in a vector:

$$\mathcal{V} = \begin{bmatrix} \sum_{j=0}^{d} [\mathsf{Q}(\lambda_1^1 + x, \cdots, \lambda_n^1 + x)]_j x^j \\ \sum_{j=0}^{d} [\mathsf{Q}(\lambda_1^2 + x, \cdots, \lambda_n^2 + x)]_j x^j \\ \vdots \\ \sum_{j=0}^{d} [\mathsf{Q}(\lambda_1^{d+1} + x, \cdots, \lambda_n^{d+1} + x)]_j x^j \end{bmatrix}.$$

By definition of the matrix $\mathcal{M}$, the following equality holds:

$$\mathcal{V} = \mathcal{M} \left[x^0\ x^1\ \cdots\ x^d\right]^T$$

From the previous technical lemma, we can assume that the matrix $\mathcal{M}$ is non singular with overwhelming probability. By computing the inverse (using the previous technical lemma), of the matrix $\mathcal{M}$ denoted $\tilde{M}$, we can rewrite the previous equality as follows:

$$(E) : \tilde{\mathcal{M}}\mathcal{V} = \left[x^0\ x^1\ \cdots\ x^d\right]^T$$

Let $\tilde{m}_{i,j}$ be the element of the i^{th} row and j^{th} column of the matrix $\tilde{\mathcal{M}}$. Using the $\tilde{m}_{i,j}$'s, we can compute the following quantities:

$$\begin{aligned} q_1 &= (\tilde{z}_0)^{\tilde{m}_{0,0}} \times \cdots \times (\tilde{z}_d)^{\tilde{m}_{0,d}} \\ &\vdots \\ q_{d-1} &= (\tilde{z}_0)^{\tilde{m}_{d,0}} \times \cdots \times (\tilde{z}_d)^{\tilde{m}_{d,d}} \end{aligned}.$$

By definition, for any k, q_k simplifies as follows:

$$g^{l_k},\ \text{with } l_k = \tilde{m}_{k,0} \cdot \sum_{j=0}^{d} [\mathsf{Q}(\lambda_1^1 + x, \cdots, \lambda_n^1 + x)]_j x^j + \cdots + \tilde{m}_{k,d} \cdot \sum_{j=0}^{d} [\mathsf{Q}(\lambda_1^{d+1} + x, \cdots, \lambda_n^{d+1} + x)]_j x^j.$$

Using equality (E), we obtain: $q_k = g^{x^{k+1}}$.
Thus, we can obtain all the power of x from 2 to d. □

The previous result shows that if we have access to an oracle that can compute a polynomial of degree d, it is possible, using $d+1$ oracle calls to obtain an oracle that compute any n^{th}-CDH-assumption with $n \leq d$. However, the proof is based on the fact that an ideal oracle always outputs the correct evaluation of the polynomial when queried. In the next section we show how to adapt the result in the more general case where we only have access to an imperfect, even adversarial oracle.

4.2 Using an adversarial oracle

In this section, we are going to prove the following result:

Theorem 2. *Given access to an adversarial perfect oracle that solves the Q-CDH-assumption with error probability bounded by ε, we can simultaneously solve instances of n^{th}-CDH-assumption for the same input g^x in all degrees from 2 to $d = \deg Q$.*
This requires $O((d+2)/\varepsilon)$ oracle calls, a computational runtime of $O(\varepsilon^{-d+2})$ and yields a success probability:

$$\frac{p\varepsilon^{d+2}}{p\varepsilon^{d+2} + 1}$$

Proof. First, let us recall that the λ_i^1 are randomly sampled (using the same notations as in the previous subsection), so the $x + \lambda_i^1$ are indistinguishable from random values. In truth, the distribution of tuples of the form $(g^{x+\lambda_i^1}, \cdots, g^{x+\lambda_i^n})$ is identical to the distribution: $(g^{r_1}, \cdots, g^{r_n})$ where $\{r_1, \cdots, r_n\} \in_R \mathbb{F}_p$. As a consequence, the oracle is not able to adapt its behavior depending on the probability distribution of our questions[1].
Thus we safely can assume that every time we submit a query to the oracle the probability that it answers correctly with probability $\geq \varepsilon$. However, if a single answer is incorrect, then our full vector of values g^{x^i} becomes incorrect (at the end of the proof the evaluations of the q_k's are all wrong). This is why we need to propose a way to test whether or not the answers of the oracle are consistent. This is summed up in the following lemma:

[1] Remember that the oracle is stateless and thus not allowed to misbehave by counting the questions and giving an answer that depends on the position of the question. It would be easy to adapt to this case, but we have not considered it.

Lemma 2 (Adversarial oracle testing).
Given a set of $d+1$ answers to randomly chosen queries from an adversarial oracle solving the Q-CDH-assumption, there exists an algorithm that tests whether all the solutions of this set are correct or not and correctly answers with probability:

$$\frac{\varepsilon^{d+1}}{\varepsilon^{d+1}+(1/p)}.$$

Assume that we have $d+1$ answers stored in a vector $\hat{\mathcal{V}}$ from the oracle and that they are all correct. As in the previous case we compute the following matrix:

$$\hat{\mathcal{M}} = \begin{bmatrix} [\mathsf{Q}(\lambda_1^1+x,\cdots,\lambda_n^1+x)]_0 & \cdots & [\mathsf{Q}(\lambda_1^1+x,\cdots,\lambda_n^1+x)]_d \\ [\mathsf{Q}(\lambda_1^2+x,\cdots,\lambda_n^2+x)]_0 & \cdots & [\mathsf{Q}(\lambda_1^2+x,\cdots,\lambda_n^2+x)]_d \\ \vdots & \vdots & \vdots \\ [\mathsf{Q}(\lambda_1^{d+1}+x,\cdots,\lambda_n^{d+1}+x)]_0 & \cdots & [\mathsf{Q}(\lambda_1^{d+1}+x,\cdots,\lambda_n^{d+1}+x)]_d \\ [\mathsf{Q}(\lambda_1^{d+2}+x,\cdots,\lambda_n^{d+2}+x)]_0 & \cdots & [\mathsf{Q}(\lambda_1^{d+2}+x,\cdots,\lambda_n^{d+2}+x)]_d \end{bmatrix}$$

Note that, compared to the previous case (when we had access to a perfect oracle), the matrix now has an extra line. And we still have the following equality:

$$\hat{\mathcal{M}} \begin{bmatrix} x^0 \\ x^1 \\ \vdots \\ x^n \end{bmatrix} = \hat{\mathcal{V}}.$$

Since this matrix has $d+2$ lines and $d+1$ columns, we known that, using simple linear algebra, we can obtain a vector $\mathcal{K}$ that is an element of the left kernel of the matrix $\hat{\mathcal{M}}$. By multiplying each part of the previous equality by the vector $\mathcal{K}$, we have:

$$\left\langle \mathcal{K} | \hat{\mathcal{V}} \right\rangle = 0.$$

Using the coordinates k_i's of the vector $\mathcal{K}$, we can check whether the oracle's answers are all consistent. Indeed, using the same notation as above, we can compute:

$$(\tilde{z}_0)^{k_0} \times \cdots \times (\tilde{z}_d)^{k_d}.$$

And the we test whether or not this value is $1 = g^0$. This can occur in two different ways. The first way occurs when the $d+1$ answer are correct and has probability ε^{d+1}. The second way occurs when the test equality is satisfied by a vector containing some random values, this occurs with probability $1/p$.
As a consequence, the *a posteriory* probability that the $d+1$ answers are correct when the test equality is satisfied is:

$$\frac{\varepsilon^{d+1}}{\varepsilon^{d+1}+(1/p)},$$

where p is the cardinality of the group. This is close to 1 for reasonable parameter choices. This concludes the proof of the previous lemma.

Remark 1. This probability is close to 1, whenever $\varepsilon^{d+1} > (1/p)$. This requires $(d+1)\ln\varepsilon$ to remain smaller (and bounded away) from $\ln p$.

In order to obtain a set of $d+1$ correct answers from the adversarial oracle we submit a large enough amount of queries to the oracle to ensure that the probability of obtaining (at least) d correct answers within the set of all the oracle answers is good enough. By "large enough amount of queries" we mean at least $c.(d+1)\times 1/\varepsilon$ where $c > 1$. With this number of queries, we are almost certain to obtain at least $d+1$ correct answers from the oracle.

The next set is to extract a subset of $d+1$ correct answers from the set of all the answers. Since there is no way to get information about the validity of an individual answer, the best approach is to use exhaustive search. Let Q_ε denote the number of correct answers, the fraction of subset of $d+1$ answer than only contains correct answer is:

$$\frac{\binom{Q_\varepsilon}{d+1}}{\binom{c.(d+1)\times\varepsilon}{d+1}}.$$

Assuming the degree d is not too large, this can be approximated by:

$$\left(\frac{1}{\varepsilon}\right)^d.$$

Thus, the cost of finding a correct subset is ε^{-d}. This is quite high, however, it is a pure computational cost that does not require any extra oracle calls.
Putting it all together, we then delete one row at random in the matrix $\tilde{\mathcal{M}}$ and assume that all the corresponding oracle's outputs are correct. We then proceed as if we had access to an ideal oracle (see lemma **??**). □

The fact that we are actually able to test the answers given by an adversarial oracle allows us to extend the random self-reducibility result of Abadi and al in [2]:

Theorem 3 (Random self-reducibility of the Q-CDH-assumption). *Given access to an adversarial oracle that solves the Q-CDH-assumption for a degree d polynomial, we can solve the Q-CDH-assumption with probability:*

$$\frac{p\varepsilon^{d+1}}{p\varepsilon^{d+1}+1}$$

using $O((d+1)/\varepsilon)$ queries and a computational runtime of $O(\varepsilon^{-(d+1)})$.

Remark 2. This probability is close to 1, whenever $\varepsilon^{d+1} > (1/p)$. This requires $(d+1)\ln\varepsilon$ to remain smaller (and bounded away) from $\ln p$.

In this section, we only considered the group setting. However, in the proof we never had to assume that the inputs or the outputs belong to the same group. Thus everything that has been stated can be straightforwardly adapted to the bilinear setting. This adaptation requires some considerations especially when dealing with pairings of type 2 or 3. The details are given in section 7.
From now on, when we use an oracle, we assume that we are given access to an almost perfect.

5 From powers to n-product

In this section, we explain how to solve the n-CDH-assumption using an almost perfect oracle that solves the n^{th}-CDH-assumption with a low probability of failure ε is close to 1.

Theorem 4. *Given access to an oracle that solves the n^{th}-CDH-assumption, we can solves the n-CDH-assumption. This requires 2^n oracle calls and needs all oracle answers to be correct.*

Proof. Once again, we state the proof and the result in the group setting. Let G be a group of prime order p and g be a generator of this group. We also assume that we have access to an oracle associated to the n^{th}-CDH-assumption with success probability $\geq \varepsilon$. We explain how to compute the quantity $g^{x_1\cdots x_n}$ knowing $(g^{x_1},\cdots,g^{x_n})$.
We first require the following technical lemma:

Lemma 3. *Let m be a monomial in $\mathbb{F}_p[X_1,\cdots,X_d]$ of degree d.*

$$\sum_{(\epsilon_1,\cdots,\epsilon_d)\in\{-1,1\}^d} \epsilon_1\cdots\epsilon_d m(\epsilon_1 X_1,\cdots,\epsilon_d X_d) = \begin{cases} 2^d X_1\cdots X_d \text{ if } m = X_1\cdots X_d \\ 0 \text{ otherwise} \end{cases}$$

The proof of this lemma is given in appendix (see appendix B).
The idea of the reduction between the n^{th}-CDH-assumption and the n-CDH-assumption is to use the previous lemma with polynomial $(X_1 + \cdots + X_d)^d$. Multiplying the outputs of 2^d oracle calls on $g^{\sum_{i=1}^{d} \epsilon_i x_i}$, we obtain the value $g^{d!\, 2^d\, x_1 \cdots x_d}$. Since $d!\, 2^d$ is invertible modulo p, we easily recover $g^{x_1 \cdots x_d}$.
Since this only works when all answers are correct, this requires ε to be close enough to 1 to have $\varepsilon^{2^d} \geq c$, for some constant $1 > c > 0$. □

It is interesting to see when the anwers can all be correct when using as oracle the algorithm from section 4. In this case, we can write:

$$\varepsilon = \frac{p\varepsilon_0^d}{p\varepsilon_0^d + 1},$$

where ε_0 is the success probability of the initial oracle for the Q-CDH-assumption.
We now consider the condition on ε_0 that arises from $\varepsilon^{2^d} \geq c$. Taking logs, we require:

$$-2^d \log 1 + (1/p)\varepsilon_0^{-d} \geq -\log(1/c) \quad \text{a sufficient condition is thus:}$$
$$(2/\varepsilon_0)^d \leq p \log(1/c) = O(p).$$

The previous reduction requires an exponential amount of queries to the oracle. It might seem a bit odd to consider such a reduction. In fact, it is not a "bad" reduction from a complexity point of view since the exponential contribution does not depend on the size of the group but on the power. Thus this reduction is acceptable.
In the next section we explain how the n-CDH-assumption reduce to the Q-CDH-assumption.

6 From n-product to polynomials

In this section we explain how it is possible to compute in the exponent any polynomial of degree d having access to an almost perfect oracle that solves the d-CDH-problem. We do this for a known polynomial Q whose coefficients are given *in the clear*. Our result is stated in the following theorem:

Theorem 5. *Let m be the number of monomials that appears in Q. Given access to an almost perfect oracle that solves the n-CDH-assumptionwith probability $\geq \varepsilon$, we build an algorithm that solves the Q-CDH-assumption for any polynomial Q of degree up to n with probability:*

$$\left(\frac{p\varepsilon^n}{p\varepsilon^n + 1}\right)^m$$

using $O(m \cdot n/\varepsilon)$ queries and a computational runtime of $O(m \cdot \varepsilon^{-n})$.

Even though this reduction need an exponential amount of queries, its exponential dependence is on the degree of the polynomial Q we want to compute and not in the cryptographic parameters (more precisely not in the size of the group). It can thus be considered as efficient.

Proof. The basic is that we can individually compute all the monomials that appears in the polynomial Q, before combining them with their respective coefficients. The monomials of Q are of the form:

$$\alpha_{i_{1,j},\cdots,i_{2,j}} \cdot x_1^{i_{1,j}} \cdots x_n^{i_{n,j}}, \text{ with } i_{1,j} + \cdots + i_{n,j} \leq d.$$

As the values $\alpha_{i_{1,j},\cdots,i_{2,j}}$ are explicitly known we only have to be able to compute the product $x_1^{i_{1,j}} \cdots x_n^{i_{n,j}}$ where the $i_{i,j}$ are known. With an ideal oracle, it would suffices to query it on the following input:

$$\underbrace{g^{x_1} \cdots g^{x_1}}_{i_{1,j}} \underbrace{g^{x_2} \cdots g^{x_2}}_{i_{2,j}} \cdots \underbrace{g^{x_n} \cdots g^{x_n}}_{i_{n,j}}.$$

However, since repetitions are extremely rare on randomly generated inputs, nothing prevents an adversarial oracle to answer these specific queries wrongly, without violating its global error bound.
Thus, in order obtain the product, we need to randomize the inputs and derandomize the answer of the oracle. It can be done by sampling independently d random non zero values $\mu_1, \cdots \mu_n$ in $\mathbb{F}_p$ and then querying the oracle on the following input:

$$\underbrace{g^{\mu_1 x_1} \cdots g^{\mu_{i_{1,j}} x_1}}_{i_{1,j}} \underbrace{g^{\mu_{i_{1,j}+1} x_2} \cdots g^{\mu_{i_{2,j}} x_2}}_{i_{2,j}} \cdots \underbrace{g^{\mu_{i_{d-1,j}+1} x_n} \cdots g^{\mu_{i_{d,j}} x_n}}_{i_{n,j}}.$$

From the answer y of the oracle, one can compute $y^{1/\prod_{i=1}^{d} \mu_i}$ and obtain the expected value.
Finally, using the result of theorem 3, we can correctly evaluate all the monomials of Q. And thus we can build an algorithm that actually compute correctly the polynomial Q from all its monomials. □

Note that a product oracle for degree d can easily be used to compute monomials of lower degree. Indeed, it suffices to replace some of the variables in the oracle call by randomly sampled constants. This allow us to state the following theorem:

Theorem 6. *In the group setting, all the n-CDH-assumptions are equivalent (w.r.t. n).*

Proof. The proof of the previous theorem relies on the fact that all those assumptions are actually equivalent to the CDH-assumption. Indeed, if we assume that we have access to an oracle solving the n-CDH-assumption, with $n \geq 2$, we can compute the CDH-assumption easily by calling the oracle on the following instance:

$$x_1 = x, x_2 = y, x_3 = \lambda_3, \cdots, x_n = \lambda_n, \text{ with } \lambda_3, \cdots, \lambda_n \text{ being randomly sampled.}$$

Using the same method as the one presented above, we can retrieve the result: g^{xy}.
If we assume that we have access to an oracle solving the CDH-assumption, by repeatedly calling it we can compute the n-CDH-assumption. First, we query the oracle with instance (g^{x_1}, g^{x_2}) and retrieve $(g^{x_1 x_2})$. Then we query the oracle with instance $(g^{x_1 x_2}, g^{x_3})$ and retrieve $(g^{x_1 x_2 x_3})$.
By repeating the process, we can compute $g^{x_1 x_2 \cdots x_n}$. Since this proof does not rely on the value of n (we only require that $n \geq 2$), we can conclude that all the n-CDH-assumptionare equivalent. □

This result should be compared to the result of Maurer and Wolf [21], regarding the equivalence of the CDH-assumption and the discrete logarithm problem. Indeed, this equivalence can be used to show that all the assumptions we are considering are easy when the CDH-assumption is. However, this only covers one direction and the reduction is not explicitly constructive since it relies on the assumption that an auxiliary elliptic curve with smooth cardinality exists and can be found.
In the group setting, as the input and the output of an oracle belong to the same set (actually the same group), we can reuse an answer of an oracle in order to compute more terms: we can iterate the oracle queries and thus from a computational Diffie-Hellman oracle, we can obtain a polynomial of any degree. Whereas, in the bilinear context it is not possible to reuse the output of the oracle as a query but the different relations that we stated in the previous theorems can be adapted to the bilinear context. This adaptation is described in the next section.

7 The bilinear setting

All the previous sections focused on the group setting. We now show how our results can be adapted to the bilinear context. The main difference between the two settings is that in the group setting it is possible to actually reuse the output of an oracle to build another query whereas in the pairing setting this is not possible. Indeed, oracle answers do not belong to the same group as queries.
The type of the pairing we are considering is also important, indeed with a type 1 or 2 pairing it is possible to perform some additional operations which are not permitted with a type 3 pairing.

Pairing of type 1

For pairings of type 1, theorems 3 and 2 and 4 and 5 can be straightforwardly adapted (the proofs of those theorems are omitted in this paper but will be provided in an extended version). Meaning that in any case, we will have the following relationships:

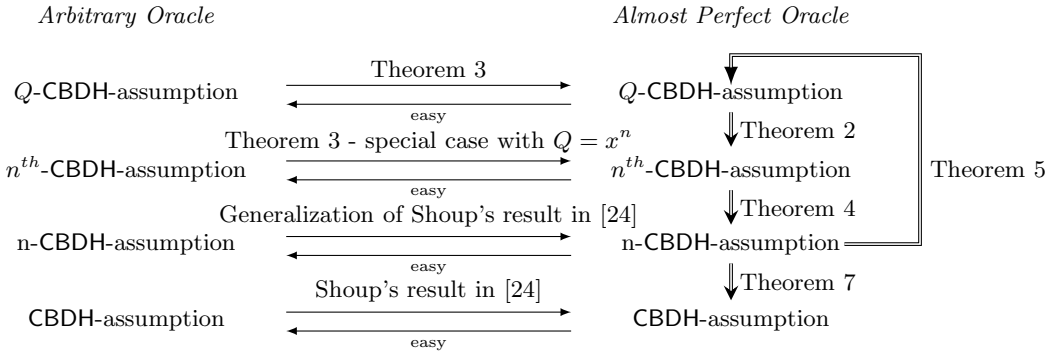

Fig. 2: Security assumptions relations (Bilinear setting)

Theorem 6 can not be stated because in the proof, we need to reuse an oracle output as input. However, we can state that:

Theorem 7. *Given access to an oracle that solves the n-CDH-assumption, we can solve any d-CDH-assumption where $d \leq n$.*

This theorem induces a hierarchy within the different class of assumptions. Indeed, the higher n is the harder the assumption is. This can be summarized as follows:

$$\text{CBDH-assumption} \leq \text{n-CBDH-assumption}(n = 4) \leq \cdots \leq \text{n-CBDH-assumption}(n = d).$$

Even though there is a hierarchy, all the different assumptions reduce to the classical CBDH-assumption.

Pairing of type 2 and 3

For pairing of type 2 and 3, once again, theorems 3 and 2 and 4 and 5 can be straightforwardly adapted (the proofs of those theorems are omitted in this paper but will be provided in an extended version). In both type of pairings, some computations cannot be done.
For pairing of type 2, assume that we have an oracle that on input $g_1^{x_1}, g_1^{x_2}, \cdots, g_1^{x_{n_1}}, g_2^{y_1}, g_2^{y_2}, \cdots, g_2^{y_{n_2}}$ outputs $h^{x_1 \cdots x_{n_1} y_1 \cdots y_{n_2}}$. Using this oracle, it is not possible to build an algorithm that can compute $h^{x_1 \cdots x_{n_1} y_1 \cdots y_{n_2}}$ on input $g_1^{x_1}, g_1^{x_2}, \cdots, g_1^{x_{n_1}}, g_1^{y_1}, g_2^{y_2}, \cdots, g_2^{y_{n_2}}$.
This impossibility introduces another kind of hierarchy based on what inputs can be derived from others. On Figure 3, we only represent the inputs of the assumption assuming we are computing the n-CBDH-assumption(on each line n is decreasing by one).
For pairing of type 3, there is no isomorphism between the two base groups and thus some more queries cannot be derived from others. This implies a different hierarchy summarized on Figure 4.

8 Conclusion

In our paper, we focus on relationships between Diffie-Hellman like security assumptions. In section 4 we explain how it is possible to use an adversarial oracle solving Q-CDH-assumptionto build an algorithm that almost certainly solves the Q-CDH-assumption. Using this algorithm, we show how to solve the n^{th}-CDH-assumption.

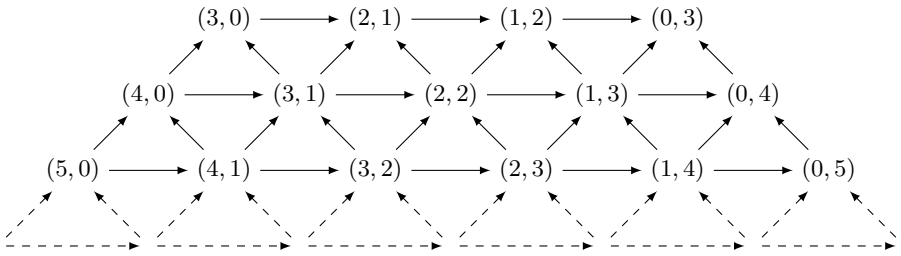

Fig. 3 *Hierarchy for type 2 pairings* − The different points of the lattice are labeled with regard to the number of variable that are in a given group. The first coordinate corresponds to the number of variables given in the group $\mathbb{G}_1$ and the second coordinate corresponds to the number of variables given in the group $\mathbb{G}_2$. As an example, the following entry $g_1^{x_1}, \cdots, g_1^{x_{n_1}}, g_2^{y_1}, \cdots g_2^{y_{n_2}}$ corresponds to the point (n_1, n_2).
The arrows define the different reduction that can be performed from a given entry to another. The horizontal arrows represent the application of the φ morphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ while the others arrows represent the fact that by setting a variable to a constant, it is possible to obtain an oracle for less entries.
The relationships between the different assumptions allows us to consider that the entries of the lattice can be submitted to any of those assumptions.

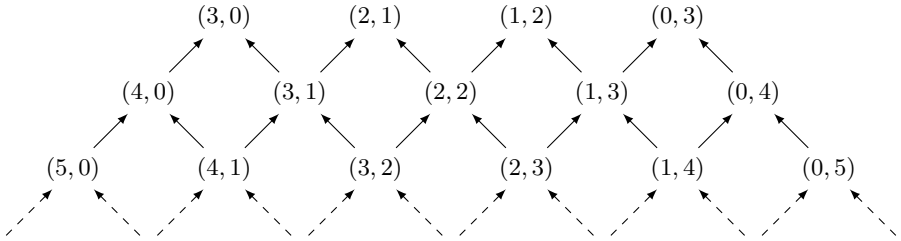

Fig. 4 *Hierarchy for type 3 pairings* − As in the previous figure (see figure 3), the different points of the lattice are labeled with regard to the number of variable that are in a given group.
The arrows define the different reduction that can be performed from a given entry to another. Note that there is no more horizontal arrows since there is no φ morphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ while the others arrows remains.

Then in section 5, we show how an oracle solving the n^{th}-CDH-assumption can be used to solve the n-CDH-assumption. Finally, in section 6, we show how an oracle that solves the n-CDH-assumption can be used to build an algorithm solving the CDH-assumptionand conversely how an oracle that solves the CDH-assumption can be used to solve the n-CDH-assumption. All the previous results leading to the following result: in the group setting, every hypothesis that we have introduce are in fact equivalent in terms of security see figure 1.

We examine the pairing setting in section 7. Although almost every results of the group setting can be easily adapted, it was not possible to prove that the CBDH-assumption and the n-CBDH-assumption are equivalent. We only prove that the n-CBDH-assumption is harder than the CBDH-assumption. This strict distinction between the CBDH-assumption and the n-CBDH-assumption introduces a hierarchy between the different pairing assumptions: the greater the number of variable is the harder the assumption is see figure 2. It was also necessary to distinguish between the different kind of pairings because some assumptions can not be reduce to others. The pairing hierarchy is quite surprising with regard to the equivalence we face in the group setting. But it is encouraging since it underlines the importance of looking over the different instantiations of the *Uber assumption*.

References

1. *The Uber-Assumption Family – A Unified Complexity Framework for Bilinear Groups*, volume 5209 of *Lecture Notes in Computer Science*. Berlin: Springer-Verlag, 2008. Available at `http://www.cs.stanford.edu/~xb/pairing08/`.
2. Martín Abadi, Joan Feigenbaum, and Joe Kilian. On hiding information from an oracle. *J. Comput. Syst. Sci.*, 39(1):21–50, 1989.
3. Feng Bao, Robert H. Deng, and Huafei Zhu. Variations of diffie-hellman problem. In Sihan Qing, Dieter Gollmann, and Jianying Zhou, editors, *ICICS*, volume 2836 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2003.
4. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. *IACR Cryptology ePrint Archive*, 2005:133, 2005.
5. Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. Computer-aided security proofs for the working cryptographer. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 71–90. Springer, 2011.
6. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.
7. I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.
8. Bruno Blanchet. Security protocol verification: Symbolic and computational models. In Pierpaolo Degano and Joshua D. Guttman, editors, *POST*, volume 7215 of *Lecture Notes in Computer Science*, pages 3–29. Springer, 2012.
9. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. Cryptology ePrint Archive, Report 2005/015, 2005. `http://eprint.iacr.org/`.
10. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.
11. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *J. Cryptology*, 17(4):297–319, 2004.
12. Mike Burmester, Yvo Desmedt, and Jennifer Seberry. Equitable key escrow with limited time span (or, how to enforce time expiration cryptographically). In Kazuo Ohta and Dingyi Pei, editors, *ASIACRYPT*, volume 1514 of *Lecture Notes in Computer Science*, pages 380–391. Springer, 1998.
13. Henri Cohen and Gerhard Frey, editors. *Handbook of elliptic and hyperelliptic curve cryptography*. CRC Press, 2005.
14. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
15. Joan Feigenbaum and Lance Fortnow. On the random-self-reducibility of complete sets. In *Structure in Complexity Theory Conference*, pages 124–132, 1991.
16. David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23(2):224–280, 2010.
17. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
18. SudhirR. Ghorpade and Gilles Lachaud. Number of solutions of equations over finite fields and a conjecture of lang and weil. In A.K. Agarwal, BruceC. Berndt, ChristianF. Krattenthaler, GaryL. Mullen, K. Ramachandra, and Michel Waldschmidt, editors, *Number Theory and Discrete Mathematics*, Trends in Mathematics, pages 269–291. Birkhuser Basel, 2002.
19. A Joux. A one round protocol for tripartite Diffie-Hellman. In W Bosma, editor, *Algorithmic Number Theory*, volume 1838, pages 385–394. Springer, 2000.
20. Serge Lang and André Weil. Number of points of varieties in finite fields. *American Journal of Mathematics*, 76:819–827, 1954.
21. Ueli M. Maurer and Stefan Wolf. Diffie-Hellman oracles. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 268–282. Springer, 1996.
22. Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.
23. Victor S. Miller. The Weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–261, 2004.
24. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.
25. Terry Tao. The Lang-Weil bound, 2012.

A Proof of lemma 1

In this section of the appendix, we prove that the matrix

$$\mathcal{M} = \begin{bmatrix} [\mathsf{Q}(\lambda_1^1 + x, \cdots, \lambda_n^1 + x)]_0 & \cdots & [\mathsf{Q}(\lambda_1^1 + x, \cdots, \lambda_n^1 + x)]_d \\ [\mathsf{Q}(\lambda_1^2 + x, \cdots, \lambda_n^2 + x)]_0 & \cdots & [\mathsf{Q}(\lambda_1^2 + x, \cdots, \lambda_n^2 + x)]_d \\ \vdots & \vdots & \vdots \\ [\mathsf{Q}(\lambda_1^{d-1} + x, \cdots, \lambda_n^{d-1} + x)]_0 & \cdots & [\mathsf{Q}(\lambda_1^{d-1} + x, \cdots, \lambda_n^{d-1} + x)]_d \end{bmatrix}$$

is non singular with probability at least $1 - \frac{d+1}{p}$ as long as the λ_i's are randomly sampled in $\mathbb{F}_p$.

Proof. The proof of this lemma rely on the fact that the determinant can be seen as an algebraic variety over $\mathbb{F}_p$. Thus as long as it is not null, the number of points N on this variety of degree d and of dimension $n(d+1)-1$ can be bounded by (see [18] and [20]):

$$|N - \pi| \leq (\mathsf{d}-1)(\mathsf{d}-2)p^{n(d+1)-3/2} + Cp^{n(d+1)-2}$$

where C is a constant and $\pi = \sum_{i=0}^{n(d+1)-1} p^i$. However, this bound is valid when the considered variety is irreducible.
From Tao's webpage [25], we find that the following equality:

$$N = \left(c(V) + O_M(p^{-1/2})\right) p^{n(d+1)-1}.$$

Where $c(V)$ is the number of geometrically irreducible components of the variety V and M is the maximum complexity of the V.
In our particular case, we cannot evaluate accurately $c(V)$ but as our variety is of degree d, we can ensure that $c(V) \leq d$ and thus we obtain the following inequality:

$$N = \left(d + O_M(p^{-1/2})\right) p^{n(d+1)-1}.$$

Our goal is to bound the probability for a point to be on this variety, we obtain directly from the previous inequality that:

$$(E)\frac{N}{p^{n(d+1)}} \leq \frac{d}{p} + \frac{O_M(p^{-1/2})}{p^{n(d+1)}} \leq \frac{d+1}{p}.$$

With reasonnable parameters choices, i.e., a prime p that guarantees the security of the discrete log for exemple, d is very small compared to p thus the probability for a random point to be on this variety is $O(d+1/p)$.
This allows us to state that, once again if the algebraic variety is not the full space, if we sample randomly the λ_i's, then the probability for the matrix $\mathcal{M}$ to be non singular is at least:

$$1 - \frac{d+1}{p}$$

However, we did not prove that this algebraic variety was not the full space and this is the second step of the proof: proving that there is at least one point that is not on this variety.
In order to do so we consider the $d+1$ constants: $\beta_0, \cdots \beta_{d+1}$. We set the β_i's to be non zero and all different (for all $i,\ j,\ \beta_i \neq \beta_j$). And we consider the following matrix :

$$\begin{bmatrix} [\mathsf{Q}(x+\beta_1, \cdots, x+\beta_1)]_0 & \cdots & [\mathsf{Q}(x+\beta_1, \cdots, x+\beta_1)]_d \\ [\mathsf{Q}(x+\beta_2, \cdots, x+\beta_2)]_0 & \cdots & [\mathsf{Q}(x+\beta_2, \cdots, x+\beta_2)]_d \\ \vdots & \vdots & \vdots \\ [\mathsf{Q}(x+\beta_{d+1}, \cdots, x+\beta_{d+1})]_0 & \cdots & [\mathsf{Q}(x+\beta_{d+1}, \cdots, x+\beta_{d+1})]_d \end{bmatrix}$$

Every coefficient of this matrix is a polynomial in the β_i and note that furthermore, on a given column, all the polynomials are actually the same.
Let's focus on one line of the matrix, the i^{th} line denoted $L_i = [\mathsf{Q}(x+\beta_i, \cdots, x+\beta_i)]_0, \cdots, [\mathsf{Q}(x+\beta_i, \cdots, x+\beta_i)]_d$. The j^{th} (with index starting at 0) element of this row vector is a polynomial in β_i and is of degree at most $d-j$. We need to distinguish between two different cases.

case 1:
Assume that all those polynomials are actually of degree exactly $d-j$. If we consider the j^{th} element, it is a polynomial in β_i and thus can be written as:

$$L_i[j] = \sum_{l=0}^{j} \gamma_l^j \beta_i^l.$$

Let's perform some operation on the matrix that will not change the fact that its determinant is non-zero. First, as the polynomial Q can be safely assume to be non zero, the last term of each line is actually a non-zero constant γ^d and thus we can set

$$C_d \leftarrow \frac{1}{\gamma^d} C_d.$$

Then, starting from $j = d-1$ down to $j = 0$, we do:

$$C_j \leftarrow \frac{1}{\gamma^j} \left(C_j - \sum_{l=0}^{j-1} \gamma_l^j C_l \right).$$

After all those operations, we are left with a vandermonde matrix. As we assumed the β_i's to be all different, the determiant of the matrix is not null and thus the matrix is non singular.

case 2:
If all the polynomials in the β_i's are not of degree exactly $d-j$ we use n additionnal values: $a_1, \cdots, a_n$ and consider the following matrix:

$$M = \begin{bmatrix} [\mathsf{Q}(x+\beta_1 a_1, \cdots, x+\beta_1 a_n)]_0 & \cdots & [\mathsf{Q}(x+\beta_1 a_1, \cdots, x+\beta_1 a_n)]_d \\ [\mathsf{Q}(x+\beta_2 a_1, \cdots, x+\beta_2 a_n)]_0 & \cdots & [\mathsf{Q}(x+\beta_2 a_1, \cdots, x+\beta_2 a_n)]_d \\ \vdots & \vdots & \vdots \\ [\mathsf{Q}(x+\beta_{d+1} a_1, \cdots, x+\beta_{d+1} a_n)]_0 & \cdots & [\mathsf{Q}(x+\beta_{d+1} a_1, \cdots, x+\beta_{d+1} a_n)]_d \end{bmatrix}$$

As in the previous case, we can consider each coefficient $m_{i,j}$ of the matrix as a polynomial in β_i. We can rewrite the $m_{i,j}$ as follows:

$$m_{i,j} = \sum_{l=0}^{j} [m_{i,j}]_l \beta_i^l.$$

Where each of the $[m_{i,j}]_l$ can be interpreted as a polynomial in the a_i's. What we want is to be able to prove that there exists at least one n-tuple of a_i's such that

$$(\Pi) : \prod_{j=0}^{d} [m_{j,j}]_{d-j} \neq 0.$$

Remark 3. Basically we want to ensure that there exists at least one n-tuple of values such that the $d+1$ different polynomials evaluated on this n-tuple do not simultaneously vanish.

Each of the $[m_{i,j}]_l$ is of degree exactly $d-l$ by construction. And each one of them defines a variety. Using the bound from [25], we obtain that:

$$|[m_{i,j}]_l| = \left(c([m_{i,j}]_l) + O_M(p^{-1/2}) \right) p^{n-1}$$

Following our reasonning from the start of this proof we have $c([m_{i,j}]_l) \leq d-l$ and thus

$$|[m_{i,j}]_l| = \left(d - l + O_M(p^{-1/2}) \right) p^{n-1}.$$

When considering the union of all those varieties (that is the product (Π)), we obtain:

$$\left|\prod_{j=0}^{d}[m_{j,j}]_{d-j}\right| \leq p^{n-1}\left(\sum_{l=0}^{d} d-l+O_{M_l}(p^{-1/2})\right)$$

Then again, we can bound the proportion of points on this variety as follows:

$$\left|\frac{\prod_{j=0}^{d}[m_{j,j}]_{d-j}}{p^n}\right| \leq \frac{d(d+1)+1}{p}.$$

With reasonnable parameters choices, we are guarenteed that there is at least one n-tuple outside this variety. Using this n-tuple we finish the proof by performing the same kind of operations on the column (see the end of the first case). This conclude the proof of the lemma. □

B Proof of lemma 3

In this section we prove the technical lemma used in theorem 4, this lemma is quite similar to the Walsh transform for polynomial with coefficients in $\mathbb{F}_2$.

Proof. First assume that $m = X_1 \cdots X_d$, then:

$$\begin{aligned}
\sum_{(\epsilon_1,\cdots,\epsilon_d)\in\{-1,1\}^d} \epsilon_1\cdots\epsilon_d m\left(\epsilon_1 X_1,\cdots,\epsilon_d X_d\right) = \\
\sum_{(\epsilon_1,\cdots,\epsilon_d)\in\{-1,1\}^d} \epsilon_1\cdots\epsilon_d\left(\epsilon_1 X_1\cdots\epsilon_d X_d\right) = \\
\sum_{(\epsilon_1,\cdots,\epsilon_d)\in\{-1,1\}^d} \epsilon_1^2\cdots\epsilon_d^2 X_1\cdots X_d = \\
X_1\cdots X_d\cdot \sum_{(\epsilon_1,\cdots,\epsilon_d)\in\{-1,1\}^d} \epsilon_1^2\cdots\epsilon_d^2 = \\
2^d X_1\cdots X_d
\end{aligned}$$

Now assume that $m \neq X_1\cdots X_d$, this implies that there exists an $i_0 \in \{1,\cdots,d\}$ such that the valuation of X_{i_0} in m is even.

$$\begin{aligned}
\sum_{(\epsilon_1,\cdots,\epsilon_d)\in\{-1,1\}^d} \epsilon_1\cdots\epsilon_d m\left(\epsilon_1 X_1,\cdots,\epsilon_d X_d\right) = \\
\sum_{(\epsilon_1,\cdots,\epsilon_d)\in\{-1,1\}^d,\epsilon_{i_0}=1} \epsilon_1\cdots\epsilon_d m\left(\epsilon_1 X_1,\cdots,X_{i_0},\cdots,\epsilon_d X_d\right) - \\
\sum_{(\epsilon_1,\cdots,\epsilon_d)\in\{-1,1\}^d,\epsilon_{i_0}=-1} \epsilon_1\cdots\epsilon_d m\left(\epsilon_1 X_1\cdots,-X_{i_0},\cdots,\epsilon_d X_d\right) = 0
\end{aligned}$$

The last equality comes from the fact that since the valuation of X_{i_0} in m is even, we have:

$$m\left(\epsilon_1 X_1\cdots,X_{i_0},\cdots,\epsilon_d X_d\right) = m\left(\epsilon_1 X_1\cdots,-X_{i_0},\cdots,\epsilon_d X_d\right).$$

This conclude the proof. □